Rajalakshmi Engineering College

Name: Jeffery Antony J
Email: 241901041@rajalakshmi.edu.in
Roll no: 241901041
Phone: 7305663808
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 3
Total Mark : 30
Marks Obtained : 20

Section 1 : Coding

1. Problem Statement

Timothy wants to evaluate polynomial expressions for his mathematics homework. He needs a program that allows him to input the coefficients of a polynomial based on its degree and compute the polynomial's value for a given input of x. Implement a function that takes the degree, coefficients, and the value of x, and returns the evaluated result of the polynomial.

Example

Input:

degree of the polynomial = 2

coefficient of x2 = 13

coefficient of x1 = 12

coefficient of x0 = 11

x = 1

Output:

36

Explanation:

Calculate the value of 13x2: 13 * 12 = 13.

Calculate the value of 12x1: 12 * 11 = 12.

Calculate the value of 11x0: 11 * 10 = 11.

Add the values of x2, x1, and x0 together: 13 + 12 + 11 = 36.

Input Format

The first line of input consists of an integer representing the degree of the polynomial.

The second line consists of an integer representing the coefficient of x2.

The third line consists of an integer representing the coefficient of x1.

The fourth line consists of an integer representing the coefficient of x0.

The fifth line consists of an integer representing the value of x, at which the polynomial should be evaluated.

Output Format

The output is an integer value obtained by evaluating the polynomial at the given value of x.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
13
12

11
1
Output: 36

Answer

```
// You are using GCC
#include <stdio.h>
#include <math.h>

int main() {
    int degree;
    int coeff2, coeff1, coeff0;
    int x, result;

    // Read degree of the polynomial
    scanf("%d", &degree);

    // Read coefficients: x^2, x^1, x^0
    scanf("%d", &coeff2);
    scanf("%d", &coeff1);
    scanf("%d", &coeff0);

    // Read value of x
    scanf("%d", &x);

    // Evaluate polynomial: ax^2 + bx + c
    result = coeff2 * x * x + coeff1 * x + coeff0;

    // Output the result
    printf("%d\n", result);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Rani is studying polynomials in her class. She has learned about polynomial multiplication and is eager to try it out on her own. However,

she finds the process of manually multiplying polynomials quite tedious. To make her task easier, she decides to write a program to multiply two polynomials represented as linked lists.

Help Rani by designing a program that takes two polynomials as input and outputs their product polynomial. Each polynomial is represented by a linked list of terms, where each term has a coefficient and an exponent. The terms are entered in descending order of exponents.

Input Format

The first line of input consists of an integer n, representing the number of terms in the first polynomial.

The following n lines of input consist of two integers each: the coefficient and the exponent of the term in the first polynomial.

The next line of input consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers each: the coefficient and the exponent of the term in the second polynomial.

Output Format

The first line of output prints the first polynomial.

The second line of output prints the second polynomial.

The third line of output prints the resulting polynomial after multiplying the given polynomials.

The polynomials should be displayed in the format, where each term is represented as ax^b, where a is the coefficient and b is the exponent.

Refer to the sample output for the exact format.

Sample Test Case

Input: 2
2 3

3 2
2
3 2
2 1

Output: 2x^3 + 3x^2
3x^2 + 2x
6x^5 + 13x^4 + 6x^3

Answer

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int coeff;
    int exp;
    struct Node* next;
} Node;

// Function to create a new term node
Node* createNode(int coeff, int exp) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->coeff = coeff;
    newNode->exp = exp;
    newNode->next = NULL;
    return newNode;
}

// Insert term into the polynomial in descending order of exponent
void insertTerm(Node** poly, int coeff, int exp) {
    if (coeff == 0) return;

    Node* newNode = createNode(coeff, exp);

    // If list is empty or exp is greater than head
    if (*poly == NULL || (*poly)->exp < exp) {
        newNode->next = *poly;
        *poly = newNode;
        return;
    }

    Node* curr = *poly;
    Node* prev = NULL;
```

```
    while (curr && curr->exp > exp) {
        prev = curr;
        curr = curr->next;
    }

    // If exponent already exists, add coefficients
    if (curr && curr->exp == exp) {
        curr->coeff += coeff;
        if (curr->coeff == 0) { // Remove zero coefficient term
            if (prev) prev->next = curr->next;
            else *poly = curr->next;
            free(curr);
        }
        free(newNode);
    } else {
        newNode->next = curr;
        if (prev) prev->next = newNode;
        else *poly = newNode;
    }
}

// Multiply two polynomials
Node* multiplyPolynomials(Node* poly1, Node* poly2) {
    Node* result = NULL;

    for (Node* i = poly1; i != NULL; i = i->next) {
        for (Node* j = poly2; j != NULL; j = j->next) {
            int coeff = i->coeff * j->coeff;
            int exp = i->exp + j->exp;
            insertTerm(&result, coeff, exp);
        }
    }

    return result;
}

// Print the polynomial in the format ax^b + ...
void printPolynomial(Node* poly) {
    if (poly == NULL) {
        printf("0\n");
        return;
```

```
    }

    Node* curr = poly;
    while (curr) {
        printf("%dx^%d", curr->coeff, curr->exp);
        if (curr->next) printf(" + ");
        curr = curr->next;
    }
    printf("\n");
}

// Read a polynomial from input
Node* readPolynomial(int terms) {
    Node* poly = NULL;
    for (int i = 0; i < terms; i++) {
        int coeff, exp;
        scanf("%d %d", &coeff, &exp);
        insertTerm(&poly, coeff, exp);
    }
    return poly;
}

int main() {
    int n, m;

    // Read first polynomial
    scanf("%d", &n);
    Node* poly1 = readPolynomial(n);

    // Read second polynomial
    scanf("%d", &m);
    Node* poly2 = readPolynomial(m);

    // Multiply
    Node* result = multiplyPolynomials(poly1, poly2);

    // Output
    printPolynomial(poly1);
    printPolynomial(poly2);
    printPolynomial(result);

    return 0;
```

}

Status : Wrong **Marks : 0/10**

3. Problem Statement

Akila is a tech enthusiast and wants to write a program to add two polynomials. Each polynomial is represented as a linked list, where each node in the list represents a term in the polynomial.

A term in the polynomial is represented in the format ax^b, where a is the coefficient and b is the exponent.

Akila needs your help to implement a program that takes two polynomials as input, adds them, and stores the result in ascending order in a new polynomial-linked list. Write a program to help her.

Input Format

The input consists of lines containing pairs of integers representing the coefficients and exponents of polynomial terms.

Each line represents a single term, with the coefficient and exponent separated by a space.

The input for each polynomial ends with a line containing "0 0".

Output Format

The output consists of three lines representing the first, second, and resulting polynomial after the addition operation, with terms sorted in ascending order of exponents.

Each line contains terms of the polynomial in the format "coefficientx^exponent", separated by " + ".

If the resulting polynomial is zero, the output is "0".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3 4
2 3
1 2
0 0
1 2
2 3
3 4
0 0

Output: 1x^2 + 2x^3 + 3x^4
1x^2 + 2x^3 + 3x^4
2x^2 + 4x^3 + 6x^4

Answer

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int coeff;
    int exp;
    struct Node* next;
} Node;

// Function to create a new node
Node* createNode(int coeff, int exp) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->coeff = coeff;
    newNode->exp = exp;
    newNode->next = NULL;
    return newNode;
}

// Insert term in ascending order of exponent
void insertSorted(Node** poly, int coeff, int exp) {
    if (coeff == 0) return;
    Node* newNode = createNode(coeff, exp);

    if (*poly == NULL || exp < (*poly)->exp) {
        newNode->next = *poly;
        *poly = newNode;
```

```
    } else {
        Node *curr = *poly, *prev = NULL;
        while (curr && curr->exp < exp) {
            prev = curr;
            curr = curr->next;
        }
        if (curr && curr->exp == exp) {
            curr->coeff += coeff;
            free(newNode);
            if (curr->coeff == 0) { // remove if zero after addition
                if (prev) prev->next = curr->next;
                else *poly = curr->next;
                free(curr);
            }
        } else {
            newNode->next = curr;
            if (prev) prev->next = newNode;
            else *poly = newNode;
        }
    }
}

// Read a polynomial until "0 0"
Node* readPolynomial() {
    Node* poly = NULL;
    int coeff, exp;
    while (1) {
        scanf("%d %d", &coeff, &exp);
        if (coeff == 0 && exp == 0) break;
        insertSorted(&poly, coeff, exp);
    }
    return poly;
}

// Add two polynomials
Node* addPolynomials(Node* poly1, Node* poly2) {
    Node* result = NULL;
    while (poly1) {
        insertSorted(&result, poly1->coeff, poly1->exp);
        poly1 = poly1->next;
    }
    while (poly2) {
```

```
        insertSorted(&result, poly2->coeff, poly2->exp);
        poly2 = poly2->next;
    }
    return result;
}

// Print polynomial in required format
void printPolynomial(Node* poly) {
    if (!poly) {
        printf("0\n");
        return;
    }

    Node* curr = poly;
    while (curr) {
        printf("%dx^%d", curr->coeff, curr->exp);
        if (curr->next) printf(" + ");
        curr = curr->next;
    }
    printf("\n");
}

int main() {
    // Read two polynomials
    Node* poly1 = readPolynomial();
    Node* poly2 = readPolynomial();

    // Add the polynomials
    Node* sum = addPolynomials(poly1, poly2);

    // Output
    printPolynomial(poly1);
    printPolynomial(poly2);
    printPolynomial(sum);

    return 0;
}
```

Status : Correct ***Marks : 10/10***