Rajalakshmi Engineering College

Name: Divya darshini S
Email: 241501051@rajalakshmi.edu.in
Roll no: 241501051
Phone: 6383045036
Branch: REC
Department: I AIML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Janani is a tech enthusiast who loves working with polynomials. She wants to create a program that can add polynomial coefficients and provide the sum of their coefficients.

The polynomials will be represented as a linked list, where each node of the linked list contains a coefficient and an exponent. The polynomial is represented in the standard form with 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 output prints the sum of the coefficients of the polynomials.

Sample Test Case

Input: 3
2 2
3 1
4 0
3
2 2
3 1
4 0
Output: 18

Answer

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

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

Node *createNode(int coeff, int exp) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->coeff = coeff;
    newNode->exp = exp;
    newNode->next = NULL;
    return newNode;
}

Node *createPolynomial(int terms) {
    Node *head = NULL;
    Node *curr = NULL;
    for (int i = 0; i < terms; i++) {
        int coeff, exp;
        scanf("%d %d", &coeff, &exp);
        Node *newNode = createNode(coeff, exp);
        if (head == NULL) {
            head = newNode;
            curr = head;
        } else {
            curr->next = newNode;
            curr = newNode;
        }
    }
    return head;
}

Node *addPolynomials(Node *poly1, Node *poly2) {
    Node *result = NULL;
```

```
    Node *curr_result = NULL;

    while (poly1 && poly2) {
        if (poly1->exp > poly2->exp) {
            Node *newNode = createNode(poly1->coeff, poly1->exp);
            if (result == NULL) {
                result = newNode;
                curr_result = result;
            } else {
                curr_result->next = newNode;
                curr_result = newNode;
            }
            poly1 = poly1->next;
        } else if (poly1->exp < poly2->exp) {
            Node *newNode = createNode(poly2->coeff, poly2->exp);
            if (result == NULL) {
                result = newNode;
                curr_result = result;
            } else {
                curr_result->next = newNode;
                curr_result = newNode;
            }
            poly2 = poly2->next;
        } else {
            int sum_coeff = poly1->coeff + poly2->coeff;
            if (sum_coeff != 0) {
                Node *newNode = createNode(sum_coeff, poly1->exp);
                if (result == NULL) {
                    result = newNode;
                    curr_result = result;
                } else {
                    curr_result->next = newNode;
                    curr_result = newNode;
                }
            }
            poly1 = poly1->next;
            poly2 = poly2->next;
        }
    }

    while (poly1) {
        Node *newNode = createNode(poly1->coeff, poly1->exp);
```

```
        if (result == NULL) {
            result = newNode;
            curr_result = result;
        } else {
            curr_result->next = newNode;
            curr_result = newNode;
        }
        poly1 = poly1->next;
    }

    while (poly2) {
        Node *newNode = createNode(poly2->coeff, poly2->exp);
        if (result == NULL) {
            result = newNode;
            curr_result = result;
        } else {
            curr_result->next = newNode;
            curr_result = newNode;
        }
        poly2 = poly2->next;
    }

    return result;
}

int sumCoefficients(Node *poly) {
    int total_sum = 0;
    Node *curr = poly;
    while (curr) {
        total_sum += curr->coeff;
        curr = curr->next;
    }
    return total_sum;
}

void freePolynomial(Node *poly) {
    Node *curr = poly;
    while (curr) {
        Node *temp = curr;
        curr = curr->next;
        free(temp);
    }
```

```
}

int main() {
    int n, m;
    scanf("%d", &n);
    Node *poly1 = createPolynomial(n);

    scanf("%d", &m);
    Node *poly2 = createPolynomial(m);

    Node *result_poly = addPolynomials(poly1, poly2);
    int sum_of_coeffs = sumCoefficients(result_poly);
    printf("%d\n", sum_of_coeffs);

    freePolynomial(poly1);
    freePolynomial(poly2);
    freePolynomial(result_poly);

    return 0;
}
```

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