Rajalakshmi Engineering College

Name: Sunetra B
Email: 241801282@rajalakshmi.edu.in
Roll no: 241801282
Phone: 9150958405
Branch: REC
Department: I AI & DS FD
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 3
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

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

typedef struct Node {
    int coefficient;
    int exponent;
    struct Node* next;
} Node;

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

void insertTerm(Node** head, int coefficient, int exponent) {
    Node* newNode = createNode(coefficient, exponent);
```

```
    if (*head == NULL || (*head)->exponent < exponent) {
        newNode->next = *head;
        *head = newNode;
    } else {
        Node* temp = *head;
        while (temp->next != NULL && temp->next->exponent > exponent) {
            temp = temp->next;
        }
        if (temp->next != NULL && temp->next->exponent == exponent) {
            temp->next->coefficient += coefficient;
            free(newNode);
        } else {
            newNode->next = temp->next;
            temp->next = newNode;
        }
    }
}

Node* readPolynomial(int n) {
    Node* head = NULL;
    int coefficient, exponent;
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &coefficient, &exponent);
        insertTerm(&head, coefficient, exponent);
    }
    return head;
}

int sumOfCoefficients(Node* head) {
    int sum = 0;
    Node* temp = head;
    while (temp != NULL) {
        sum += temp->coefficient;
        temp = temp->next;
    }
    return sum;
}

void freePolynomial(Node* head) {
    Node* temp;
    while (head != NULL) {
        temp = head;
```

```
        head = head->next;
        free(temp);
    }
}

int main() {
    int n, m;

    scanf("%d", &n);
    Node* poly1 = readPolynomial(n);

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

    Node* result = NULL;
    Node* temp = poly1;
    while (temp != NULL) {
        insertTerm(&result, temp->coefficient, temp->exponent);
        temp = temp->next;
    }
    temp = poly2;
    while (temp != NULL) {
        insertTerm(&result, temp->coefficient, temp->exponent);
        temp = temp->next;
    }

    int resultSum = sumOfCoefficients(result);

    printf("%d\n", resultSum);

    freePolynomial(poly1);
    freePolynomial(poly2);
    freePolynomial(result);

    return 0;
}
```

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