

Rajalakshmi Engineering College

Name: srivatsen s
Email: 240701534@rajalakshmi.edu.in
Roll no: 2116240701534
Phone: 9042122714
Branch: REC
Department: I CSE FE
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 2
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

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

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

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

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

```
void insertTerm(Node** head, int coef, int exp) {  
    Node* newNode = createNode(coef, exp);  
    if (*head == NULL) {  
        *head = newNode;  
    } else {
```

```

Node* current = *head;
while (current->next != NULL) {
    current = current->next;
}
current->next = newNode;
}
}

```

```

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

    while (poly1 != NULL && poly2 != NULL) {
        if (poly1->exponent == poly2->exponent) {
            int sumCoefficients = poly1->coefficient + poly2->coefficient;
            if (sumCoefficients != 0) {
                insertTerm(&result, sumCoefficients, poly1->exponent);
                if (current == NULL) {
                    current = result;
                } else {
                    current = current->next;
                }
            }
            poly1 = poly1->next;
            poly2 = poly2->next;
        } else if (poly1->exponent > poly2->exponent) {
            insertTerm(&result, poly1->coefficient, poly1->exponent);
            if (current == NULL) {
                current = result;
            } else {
                current = current->next;
            }
            poly1 = poly1->next;
        } else {
            insertTerm(&result, poly2->coefficient, poly2->exponent);
            if (current == NULL) {
                current = result;
            } else {
                current = current->next;
            }
            poly2 = poly2->next;
        }
    }
}

```

```
}  
while (poly1 != NULL) {  
    insertTerm(&result, poly1->coefficient, poly1->exponent);  
    if (current == NULL) {  
        current = result;  
    } else {  
        current = current->next;  
    }  
    poly1 = poly1->next;  
}
```

```
while (poly2 != NULL) {  
    insertTerm(&result, poly2->coefficient, poly2->exponent);  
    if (current == NULL) {  
        current = result;  
    } else {  
        current = current->next;  
    }  
    poly2 = poly2->next;  
}
```

```
return result;  
}
```

```
void printPolynomial(Node* poly) {  
    if (poly == NULL) {  
        printf("0\n");  
        return;  
    }
```

```
while (poly != NULL) {  
    if (poly->coefficient != 0) {  
        if (poly->exponent > 1) {  
            printf("%dx^%d", poly->coefficient, poly->exponent);  
        } else if (poly->exponent == 1) {  
            printf("%dx", poly->coefficient);  
        } else {  
            printf("%d", poly->coefficient);  
        }  
    }
```

```
    if (poly->next != NULL && poly->next->coefficient > 0) {
```

```
        printf(" + ");
    }
}
poly = poly->next;
}

printf("\n");
}
```

```
void freeLinkedList(Node* head) {
    Node* current = head;
    while (current != NULL) {
        Node* temp = current;
        current = current->next;
        free(temp);
    }
}
```

```
int main() {
    Node* poly1 = NULL;
    Node* poly2 = NULL;

    int num_terms_poly1, num_terms_poly2;

    scanf("%d", &num_terms_poly1);

    for (int i = 0; i < num_terms_poly1; i++) {
        int coef, exp;
        scanf("%d%d", &coef, &exp);
        insertTerm(&poly1, coef, exp);
    }

    scanf("%d", &num_terms_poly2);

    for (int i = 0; i < num_terms_poly2; i++) {
        int coef, exp;
        scanf("%d%d", &coef, &exp);
        insertTerm(&poly2, coef, exp);
    }

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

```
int sum = 0;
Node* current = result;
while (current != NULL) {
    sum += current->coefficient;
    current = current->next;
}
printf("%d", sum);

freeLinkedList(poly1);
freeLinkedList(poly2);
freeLinkedList(result);
return 0;
}
```

Status : Correct

Marks : 10/10