Rajalakshmi Engineering College

Name: Ritesh Sivakumar
Email: 240701427@rajalakshmi.edu.in
Roll no: 240701427
Phone: 9342061449
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 20

Section 1 : Coding

1. Problem Statement

Keerthi is a tech enthusiast and is fascinated by polynomial expressions. She loves to perform various operations on polynomials.

Today, she is working on a program to multiply two polynomials and delete a specific term from the result.

Keerthi needs your help to implement this program. She wants to take the coefficients and exponents of the terms of the two polynomials as input, perform the multiplication, and then allow the user to specify an exponent for deletion from the resulting polynomial, and display the result.

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 representing the coefficient and the exponent of the term in the first polynomial.

The next line 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 representing the coefficient and the exponent of the term in the second polynomial.

The last line consists of an integer, representing the exponent of the term that Keerthi wants to delete from the multiplied polynomial.

Output Format

The first line of output displays the resulting polynomial after multiplication.

The second line displays the resulting polynomial after deleting the specified term.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
2 2
3 1
4 0
2
1 2
2 1
2
Output: Result of the multiplication: 2x^4 + 7x^3 + 10x^2 + 8x
Result after deleting the term: 2x^4 + 7x^3 + 8x

Answer

```
#include <stdio.h>

typedef struct {
    int coeff;
```

```
    int exp;
} Term;

void multiplyPolynomials(Term poly1[], int n, Term poly2[], int m, Term result[], int
*size) {
    Term tempResult[25];
    int tempSize = 0;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            tempResult[tempSize].coeff = poly1[i].coeff * poly2[j].coeff;
            tempResult[tempSize].exp = poly1[i].exp + poly2[j].exp;
            tempSize++;
        }
    }

    for (int i = 0; i < tempSize; i++) {
        for (int j = i + 1; j < tempSize; j++) {
            if (tempResult[i].exp == tempResult[j].exp) {
                tempResult[i].coeff += tempResult[j].coeff;
                for (int k = j; k < tempSize - 1; k++) {
                    tempResult[k] = tempResult[k + 1];
                }
                tempSize--;
                j--;
            }
        }
    }

    *size = tempSize;
    for (int i = 0; i < tempSize; i++) {
        result[i] = tempResult[i];
    }
}

void printPolynomial(Term poly[], int size, const char *message) {
    printf("%s: ", message);
    for (int i = 0; i < size; i++) {
        if (poly[i].exp == 0) {
            printf("%d", poly[i].coeff);
        } else if (poly[i].exp == 1) {
            printf("%dx", poly[i].coeff);
```

```
        } else {
            printf("%dx^%d", poly[i].coeff, poly[i].exp);
        }
        if (i != size - 1) {
            printf(" + ");
        }
    }
    printf("\n");
}

void deleteTerm(Term poly[], int *size, int expToDelete) {
    int newSize = 0;
    Term newPoly[*size];

    for (int i = 0; i < *size; i++) {
        if (poly[i].exp != expToDelete) {
            newPoly[newSize++] = poly[i];
        }
    }

    *size = newSize;
    for (int i = 0; i < newSize; i++) {
        poly[i] = newPoly[i];
    }
}

int main() {
    int n, m, expToDelete;

    scanf("%d", &n);
    Term poly1[n];
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &poly1[i].coeff, &poly1[i].exp);
    }

    scanf("%d", &m);
    Term poly2[m];
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &poly2[i].coeff, &poly2[i].exp);
    }

    Term result[25];
```

```
    int resultSize;
    multiplyPolynomials(poly1, n, poly2, m, result, &resultSize);
    printPolynomial(result, resultSize, "Result of the multiplication");

    scanf("%d", &expToDelete);
    deleteTerm(result, &resultSize, expToDelete);
    printPolynomial(result, resultSize, "Result after deleting the term");

    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>

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

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

```
}

void insertNode(struct Node** head, int coefficient, int exponent) {
    struct Node* newNode = createNode(coefficient, exponent);

    if (*head == NULL || exponent > (*head)->exponent) {
        newNode->next = *head;
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL && temp->next->exponent >= exponent) {
            temp = temp->next;
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }
}

void addSamePowerCoefficients(struct Node* head) {
    struct Node* current = head;
    while (current != NULL && current->next != NULL) {
        if (current->exponent == current->next->exponent) {
            current->coefficient += current->next->coefficient;
            struct Node* temp = current->next;
            current->next = temp->next;
            free(temp);
        } else {
            current = current->next;
        }
    }
}

struct Node* multiplyPolynomials(struct Node* poly1, struct Node* poly2) {
    struct Node* result = NULL;
    struct Node* tempPoly1 = poly1;

    while (tempPoly1 != NULL) {
        struct Node* tempPoly2 = poly2;
        while (tempPoly2 != NULL) {
            int coeff = tempPoly1->coefficient * tempPoly2->coefficient;
            int exp = tempPoly1->exponent + tempPoly2->exponent;
            insertNode(&result, coeff, exp);
```

```
            tempPoly2 = tempPoly2->next;
        }
        tempPoly1 = tempPoly1->next;
    }

    addSamePowerCoefficients(result);

    return result;
}

void displayPolynomial(struct Node* head) {
    struct Node* temp = head;
    int isFirst = 1;
    while (temp != NULL) {
        if (temp->coefficient != 0) {
            if (!isFirst && temp->coefficient > 0) {
                printf(" + ");
            }
            if (temp->exponent == 0) {
                printf("%d", temp->coefficient);
            } else if (temp->exponent == 1) {
                printf("%dx", temp->coefficient);
            } else {
                printf("%dx^%d", temp->coefficient, temp->exponent);
            }
            isFirst = 0;
        }
        temp = temp->next;
    }
    printf("\n");
}

void freePolynomial(struct Node* head) {
    while (head != NULL) {
        struct Node* temp = head;
        head = head->next;
        free(temp);
    }
}

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

```
    struct Node* poly2 = NULL;
    struct Node* product = NULL;

    int coefficient, exponent;
    int numTerms1;

    scanf("%d", &numTerms1);

    for (int i = 0; i < numTerms1; i++) {
        scanf("%d %d", &coefficient, &exponent);
        insertNode(&poly1, coefficient, exponent);
    }

    int numTerms2;

    scanf("%d", &numTerms2);
    for (int i = 0; i < numTerms2; i++) {
        scanf("%d %d", &coefficient, &exponent);
        insertNode(&poly2, coefficient, exponent);
    }

    product = multiplyPolynomials(poly1, poly2);
    displayPolynomial(poly1);
    displayPolynomial(poly2);
    displayPolynomial(product);

    freePolynomial(poly1);
    freePolynomial(poly2);
    freePolynomial(product);

    return 0;
}
```

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

3. Problem Statement

Hasini is studying polynomials in her class. Her teacher has introduced a new concept of two polynomials using linked lists.

The teacher provides Hasini with a program that takes two polynomials as input, represented as linked lists, and then displays them together. The polynomials are simplified and should be displayed in the format ax^b, where a is the coefficient and b is the exponent.

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 polynomials should be displayed in the format ax^b, where a is the coefficient and b is the exponent.

Refer to the sample output for formatting specifications.

Sample Test Case

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

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

Answer

-

Status : Skipped ***Marks : 0/10***