# Rajalakshmi Engineering College

Name: Piraisoodan R
Email: 240701384@rajalakshmi.edu.in
Roll no: 240701384
Phone: 8056892546
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 : 20.5

## 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>
#include <stdlib.h>

#define MAX_TERMS 100
```

```
typedef struct {
    int coeff;
    int exp;
} Term;

typedef struct {
    Term terms[MAX_TERMS];
    int size;
} Polynomial;

// Add term or combine like terms in order
void addTerm(Polynomial *poly, int coeff, int exp) {
    for (int i = 0; i < poly->size; i++) {
        if (poly->terms[i].exp == exp) {
            poly->terms[i].coeff += coeff;
            return;
        }
    }
    poly->terms[poly->size].coeff = coeff;
    poly->terms[poly->size].exp = exp;
    poly->size++;
}

Polynomial multiplyPolynomials(Polynomial *p1, Polynomial *p2) {
    Polynomial result = { .size = 0 };
    for (int i = 0; i < p1->size; i++) {
        for (int j = 0; j < p2->size; j++) {
            int coeff = p1->terms[i].coeff * p2->terms[j].coeff;
            int exp = p1->terms[i].exp + p2->terms[j].exp;
            addTerm(&result, coeff, exp);  // Preserve order
        }
    }
    return result;
}

void deleteTerm(Polynomial *poly, int exp) {
    for (int i = 0; i < poly->size; i++) {
        if (poly->terms[i].exp == exp) {
            for (int j = i; j < poly->size - 1; j++) {
                poly->terms[j] = poly->terms[j + 1];
            }
```

```
            poly->size--;
            break;
        }
    }
}

void printPolynomial(Polynomial *poly) {
    for (int i = 0; i < poly->size; i++) {
        if (poly->terms[i].coeff == 0)
            continue;

        if (i > 0)
            printf(" + ");

        if (poly->terms[i].exp == 0)
            printf("%d", poly->terms[i].coeff);
        else if (poly->terms[i].exp == 1)
            printf("%dx", poly->terms[i].coeff);
        else
            printf("%dx^%d", poly->terms[i].coeff, poly->terms[i].exp);
    }
    printf("\n");
}

int main() {
    Polynomial p1 = { .size = 0 }, p2 = { .size = 0 };
    int n, m, coeff, exp, delExp;

    // First polynomial
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &coeff, &exp);
        addTerm(&p1, coeff, exp);
    }

    // Second polynomial
    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &coeff, &exp);
        addTerm(&p2, coeff, exp);
    }
```

```
    // Exponent to delete
    scanf("%d", &delExp);

    Polynomial result = multiplyPolynomials(&p1, &p2);

    printf("Result of the multiplication: ");
    printPolynomial(&result);

    deleteTerm(&result, delExp);

    printf("Result after deleting the term: ");
    printPolynomial(&result);

    return 0;
}
```

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

2. 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***

```
#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;
}

void insertTerm(Node** head, int coeff, int exp) {
    Node* newNode = createNode(coeff, exp);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node* temp = *head;
        while (temp->next)
            temp = temp->next;
        temp->next = newNode;
    }
}

void printPolynomial(Node* head) {
    Node* temp = head;
    while (temp) {

        if (temp->exp == 0) {
            printf("%d", temp->coeff);
        } else if (temp->exp == 1) {
            printf("%dx", temp->coeff);
        } else {
            printf("%dx^%d", temp->coeff, temp->exp);
        }


        if (temp->next && temp->next->coeff >= 0)
            printf(" + ");

        temp = temp->next;
    }
    printf("\n");
}

int main() {
    int n, m, coeff, exp;
    Node *poly1 = NULL, *poly2 = NULL;
```

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


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


    printPolynomial(poly1);
    printPolynomial(poly2);

    return 0;
}
```

***Status :*** Partially correct **Marks : 8/10**

3. Problem Statement

Lisa is studying polynomials in her class. She is learning about the multiplication of polynomials.

To practice her understanding, she wants to write a program that multiplies two polynomials and displays the result. Each polynomial is represented as a linked list, where each node contains the coefficient and exponent of a term.

Example

Input:

4 3

y

3 1

y

1 0

n

2 2

y

3 1

y

2 0

n

Output:

8x^5 + 12x^4 + 14x^3 + 11x^2 + 9x + 2

Explanation

1. Poly1: 4x^3 + 3x + 1
2. Poly2: 2x^2 + 3x + 2

Multiplication Steps:

1. Multiply 4x^3 by Poly2:
   -> 4x^3 * 2x^2 = 8x^5
   -> 4x^3 * 3x = 12x^4
   -> 4x^3 * 2 = 8x^3
2. Multiply 3x by Poly2:
   -> 3x * 2x^2 = 6x^3
   -> 3x * 3x = 9x^2
   -> 3x * 2 = 6x

3. Multiply 1 by Poly2:

-> 1 * 2x^2 = 2x^2

-> 1 * 3x = 3x

-> 1 * 2 = 2

Combine the results: 8x^5 + 12x^4 + (8x^3 + 6x^3) + (9x^2 + 2x^2) + (6x + 3x) + 2

The combined polynomial is: 8x^5 + 12x^4 + 14x^3 + 11x^2 + 9x + 2

***Input Format***

The input consists of two sets of polynomial terms.

Each polynomial term is represented by two integers separated by a space:

- The first integer represents the coefficient of the term.
- The second integer represents the exponent of the term.

After entering a polynomial term, the user is prompted to input a character indicating whether to continue adding more terms to the polynomial.

If the user inputs 'y' or 'Y', the program continues to accept more terms.

If the user inputs 'n' or 'N', the program moves on to the next polynomial.

***Output Format***

The output consists of a single line representing the resulting polynomial after multiplying the two input polynomials.

Each term of the resulting polynomial is formatted as follows:

- The coefficient and exponent are separated by 'x^' if the exponent is greater than 1.

- If the exponent is 1, only 'x' is displayed without the exponent.
- If the exponent is 0, only the coefficient is displayed.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 4 3
y
3 1
y
1 0
n
2 2
y
3 1
y
2 0
n
Output: 8x^5 + 12x^4 + 14x^3 + 11x^2 + 9x + 2

***Answer***

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.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;
}
```

```
void insertEnd(Node** head, int coeff, int exp) {
    Node* newNode = createNode(coeff, exp);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node* temp = *head;
        while (temp->next) temp = temp->next;
        temp->next = newNode;
    }
}


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

    for (Node* p1 = poly1; p1 != NULL; p1 = p1->next) {
        for (Node* p2 = poly2; p2 != NULL; p2 = p2->next) {
            int c = p1->coeff * p2->coeff;
            int e = p1->exp + p2->exp;


            Node* temp = result;
            Node* prev = NULL;
            while (temp && temp->exp > e) {
                prev = temp;
                temp = temp->next;
            }
            if (temp && temp->exp == e) {
                temp->coeff += c;
                if (temp->coeff == 0) {
                    if (prev == NULL) {
                        result = temp->next;
                        free(temp);
                    } else {
                        prev->next = temp->next;
                        free(temp);
                    }
                }
            } else {

                Node* newNode = createNode(c, e);
                if (prev == NULL) {
```

```
                newNode->next = result;
                result = newNode;
            } else {
                newNode->next = prev->next;
                prev->next = newNode;
            }
        }
    }
    return result;
}


void printPolynomial(Node* head) {
    Node* temp = head;
    int first = 1;
    while (temp) {
        if (temp->coeff == 0) {
            temp = temp->next;
            continue;
        }

        if (!first) printf(" + ");
        first = 0;

        int c = temp->coeff;
        int e = temp->exp;


        if (e == 0) {
            printf("%d", c);
        } else if (e == 1) {
            printf("%dx", c);
        } else {
            printf("%dx^%d", c, e);
        }

        temp = temp->next;
    }
    printf("\n");
}
```

```
int main() {
    Node *poly1 = NULL, *poly2 = NULL;
    int coeff, exp;
    char ch;


    do {
        scanf("%d %d", &coeff, &exp);
        insertEnd(&poly1, coeff, exp);
        scanf(" %c", &ch);
    } while (ch == 'y' || ch == 'Y');


    do {
        scanf("%d %d", &coeff, &exp);
        insertEnd(&poly2, coeff, exp);
        scanf(" %c", &ch);
    } while (ch == 'y' || ch == 'Y');


    Node* result = multiplyPolynomials(poly1, poly2);


    printPolynomial(result);


    return 0;
}
```

***Status :*** Partially correct ***Marks : 2.5/10***