Rajalakshmi Engineering College

Name: eswar rs
Email: 240801074@rajalakshmi.edu.in
Roll no: 240801074
Phone: 7845648127
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 22.5

Section 1 : Coding

1. Problem Statement

Akila is a tech enthusiast and wants to write a program to add two polynomials. Each polynomial is represented as a linked list, where each node in the list represents a term in the polynomial.

A term in the polynomial is represented in the format ax^b, where a is the coefficient and b is the exponent.

Akila needs your help to implement a program that takes two polynomials as input, adds them, and stores the result in ascending order in a new polynomial-linked list. Write a program to help her.

Input Format

The input consists of lines containing pairs of integers representing the

coefficients and exponents of polynomial terms.

Each line represents a single term, with the coefficient and exponent separated by a space.

The input for each polynomial ends with a line containing "0 0".

Output Format

The output consists of three lines representing the first, second, and resulting polynomial after the addition operation, with terms sorted in ascending order of exponents.

Each line contains terms of the polynomial in the format "coefficientx^exponent", separated by " + ".

If the resulting polynomial is zero, the output is "0".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3 4
2 3
1 2
0 0
1 2
2 3
3 4
0 0
Output: 1x^2 + 2x^3 + 3x^4
1x^2 + 2x^3 + 3x^4
2x^2 + 4x^3 + 6x^4

Answer

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

typedef struct Node{
    int coeff;
    int exp;
```

```
    struct Node* next;
}Node;

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

void insert_sorted(Node** head,int coeff,int exp) {
    Node* new_node = create_node(coeff,exp);
    if(*head == NULL ||(*head)->exp>exp) {
        new_node->next = *head;
        *head = new_node;
    }else {
        Node* current = *head;
        while(current->next != NULL && current->next->exp<exp) {
            current = current->next;
        }
        if(current->exp == exp) {
            current->coeff += coeff;
            free(new_node);
        } else {
            new_node->next = current->next;
            current->next = new_node;
        }
    }
}
Node* add_polynomials(Node*  poly1,Node* poly2) {
    Node* result = NULL;
    while(poly1 != NULL || poly2 != NULL) {
        if(poly1 != NULL && (poly2 == NULL || poly1->exp<poly2->exp)){
            insert_sorted(&result,poly1->coeff,poly1->exp);
            poly1=poly1->next;
        }else if(poly2 != NULL && (poly1 == NULL || poly2->exp<poly1->exp)) {
            insert_sorted(&result,poly2->coeff,poly2->exp);
            poly2 = poly2->next;
        } else{
            insert_sorted(&result,poly1->coeff + poly2->coeff,poly1->exp);
            poly1=poly1->next;
```

```
            poly2=poly2->next;
        }
    }
    return result;
}

void print_polynomial(Node* poly) {

    Node *temp = poly;
    int flag=0;
    while(temp!=NULL)
    {
        if(temp->coeff!=0)
            flag=1;
        temp=temp->next;
    }

    if(!flag) {
        printf("0\n");

        return;

    }
    while(poly) {
        printf("%dx^%d",poly->coeff,poly->exp);
        if(poly->next) {
            printf(" + ");
        }
        poly = poly->next;
    }
    printf("\n");
}

Node* read_polynomial() {
    Node* poly = NULL;
    while(1) {
        int coeff,exp;
        scanf("%d%d",&coeff,&exp);
        if(coeff == 0 && exp == 0)break;
        insert_sorted(&poly,coeff,exp);
    }
    return poly;
```

```
}

int main() {
    // printf("\n");
    Node* poly1 = read_polynomial();

    // printf("\n");
    Node* poly2 = read_polynomial();

    Node* result = add_polynomials(poly1,poly2);

    print_polynomial(poly1);
    print_polynomial(poly2);
    print_polynomial(result);

    return 0;
}
```

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

2. 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

```
// 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 addTerm(Node** head, int coefficient,int exponent) {
    Node* newNode = createNode(coefficient,exponent);
    if(*head == NULL) {
        *head = newNode;
    } else {
        Node* temp = *head;
```

```
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
Node* multiplyPolynomial(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 coeff = p1->coefficient * p2->coefficient;
            int exp = p1->exponent + p2->exponent;

            Node* temp = result;
            Node* prev = NULL;
            while(temp != NULL && temp->exponent > exp) {
                prev = temp;
                temp = temp->next;
            }
            if(temp!= NULL && temp->exponent == exp) {
                temp->coefficient +=coeff;
            } else {
                Node* newNode = createNode(coeff, exp);
                if(prev == NULL) {
                    newNode->next = result;
                    result = newNode;
                } else {
                    Node* newNode = createNode(coeff, exp);
                    if(prev == NULL) {
                        newNode->next = result;
                        result = newNode;
                    } else {
                        prev->next = newNode;
                        newNode->next = temp;
                    }
                }
            }
        }
    }
    return result;
}
```

```
void displayPolynomial(Node* poly) {
    Node* temp = poly;
    int first = 1;
    while(temp != NULL) {
        if(temp->coefficient != 0) {
            if(!first && 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);
            }
            first = 0;
        }
        temp = temp->next;
    }
    printf("\n");
}
int main() {
    Node* poly1 = NULL;
    Node* poly2 = NULL;
        char ch;
        int x,y;

        do{

        scanf("%d %d\n%c",&x,&y,&ch);
        addTerm(&poly1, x, y);
    }while(ch=='y'|| ch=='Y');

    do{

        scanf("%d %d\n%c",&x,&y,&ch);
        addTerm(&poly2, x, y);
    }while(ch=='y'|| ch=='Y');


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

```
    printf("");
    displayPolynomial(result);

    return 0;
}
```

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

3. Problem Statement

Hayley loves studying polynomials, and she wants to write a program to compare two polynomials represented as linked lists and display whether they are equal or not.

The polynomials are expressed as a series of terms, where each term consists of a coefficient and an exponent. The program should read the polynomials from the user, compare them, and then display whether they are equal or not.

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

Output Format

The first line of output prints "Polynomial 1: " followed by the first polynomial.

The second line prints "Polynomial 2: " followed by the second polynomial.

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

If the two polynomials are equal, the third line prints "Polynomials are Equal."

If the two polynomials are not equal, the third line prints "Polynomials are Not Equal."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2
1 2
2 1
2
1 2
2 1
Output: Polynomial 1: (1x^2) + (2x^1)
Polynomial 2: (1x^2) + (2x^1)
Polynomials are Equal.

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
typedef struct Term {
    int coefficient;
    int exponent;
    struct Term* next;

} Term;

Term* createTerm(int coefficient,int exponent) {
    Term* newTerm = (Term*)malloc(sizeof(Term));
    newTerm->coefficient = coefficient;
    newTerm->exponent = exponent;
    newTerm->next = NULL;
    return newTerm;
}

Term* readPolynomial(int n) {
```

```
    Term* head = NULL;
    Term* tail = NULL;

    for(int i=0; i<n;i++) {
        int coeff, exp;
        scanf("%d %d",&coeff, &exp);
        Term* newTerm = createTerm(coeff, exp);

        if(head == NULL) {
            head = newTerm;
        } else{
            tail->next = newTerm;
        }
        tail = newTerm;
    }

    return head;
}

void displayPolynomial(Term* head) {
    Term* current = head;
    bool first = true;
    while(current != NULL) {
        if(!first) {
            printf(" + ");
        }
        first = false;
        printf("(%dx^%d)", current->coefficient, current->exponent);
        current = current->next;
    }
}

bool comparePolynomials(Term* poly1, Term* poly2) {
    while(poly1 != NULL && poly2 != NULL) {
        if(poly1->coefficient != poly2->coefficient || poly1->exponent != poly2-
>exponent) {
            return false;
        }
        poly1 = poly1->next;
        poly2 = poly2->next;
    }
    return poly1 == NULL && poly2 == NULL;
```

```
}
int main() {
    int n,m;
    scanf("%d",&n);
    Term* polynomial1 = readPolynomial(n);
    scanf("%d",&m);
    Term* polynomial2 = readPolynomial(m);
    printf("Polynomial 1: ");
    displayPolynomial(polynomial1);
    printf("\n");

    printf("Polynomial 2: ");
    displayPolynomial(polynomial2);
    printf("\n");

    if(comparePolynomials(polynomial1, polynomial2)) {
        printf("Polynomials are Equal.\n");
    } else{
        printf("Polynomials are Not Equal.\n");
    }
    return 0;
}
```

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