Rajalakshmi Engineering College

Name: NITHISH RAJ L
Email: 240701366@rajalakshmi.edu.in
Roll no: 2116240701366
Phone: 8072719523
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 : 27.5

Section 1 : Coding

1. 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

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

```
struct Term {
    int coefficient;
    int exponent;
    struct Term* next;
};

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

void insertTerm(struct Term** poly, int coefficient, int exponent) {
    struct Term* newTerm = createTerm(coefficient, exponent);
    if (*poly == NULL || (*poly)->exponent < exponent) {
        newTerm->next = *poly;
        *poly = newTerm;
    } else {
        struct Term* temp = *poly;
        while (temp->next != NULL && temp->next->exponent > exponent) {
            temp = temp->next;
        }
        if (temp->next != NULL && temp->next->exponent == exponent) {
            temp->next->coefficient += coefficient;
            free(newTerm);
        } else {
            newTerm->next = temp->next;
            temp->next = newTerm;
        }
    }
}

int comparePolynomials(struct Term* poly1, struct Term* poly2) {
    while (poly1 != NULL && poly2 != NULL) {
        if (poly1->exponent != poly2->exponent) {
            return 0;
        }
        if (poly1->coefficient != poly2->coefficient) {
            return 0;
        }
```

```
        poly1 = poly1->next;
        poly2 = poly2->next;
    }
    return (poly1 == NULL && poly2 == NULL);
}

void displayPolynomial(struct Term* poly) {
    if (poly == NULL) {
        printf("0\n");
        return;
    }
    struct Term* temp = poly;
    while (temp != NULL) {
        if (temp->coefficient > 0 && temp != poly) {
            printf("+");
        }
        printf(" (%dx^%d) ", temp->coefficient, temp->exponent);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    struct Term* poly1 = NULL;
    struct Term* poly2 = NULL;
    int n, coef, exp;

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

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

    printf("Polynomial 1: ");
    displayPolynomial(poly1);
    printf("Polynomial 2: ");
```

```
    displayPolynomial(poly2);

    if (comparePolynomials(poly1, poly2)) {
        printf("Polynomials are equal.\n");
    } else {
        printf("Polynomials are not equal.\n");
    }

    return 0;
}
```

Status : Partially correct **Marks : 7.5/10**

2. 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>

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 addTerm(struct Node** head, int coefficient, int exponent) {
    if(coefficient==0)
    return;
    struct Node* newNode = createNode(coefficient, exponent);
    if (*head == NULL ||exponent< (*head)->exponent) {
        newNode->next = *head;
        *head = newNode;
        return;
    } else {
        struct Node* current = *head;
        while (current->next != NULL && current->next->exponent < exponent) {
            current = current->next;
        }
        if (current->next != NULL && current->next->exponent == exponent) {
            current->next->coefficient += coefficient;
            free(newNode);
            if (current->next->coefficient == 0) {
                struct Node* temp = current->next;
                current->next = current->next->next;
                free(temp);
            }
        } else {
            newNode->next = current->next;
            current->next = newNode;
        }
    }
}

void readPolynomial(struct Node** head) {
    int coefficient, exponent;
    while (1) {
        scanf("%d %d", &coefficient, &exponent);
        if (coefficient == 0 && exponent == 0) {
            break;
        }
        addTerm(head, coefficient, exponent);
    }
}

void printPolynomial(struct Node* head) {
    if (head == NULL) {
        printf("0\n");
```

```
        return;
    }
    struct Node* current = head;
    int first = 1;
    while (current != NULL) {
        if (first) {
            printf("%dx^%d", current->coefficient, current->exponent);
            first = 0;
        } else {
            if (current->coefficient >= 0) {
                printf(" + %dx^%d", current->coefficient, current->exponent);
            } else {
                printf(" + -%dx^%d", -current->coefficient, current->exponent);
            }
        }
        current = current->next;
    }
    printf("\n");
}

struct Node* addPolynomials(struct Node* poly1, struct Node* poly2) {
    struct Node* result = NULL;
    while (poly1 != NULL || poly2 != NULL) {
        if (poly1 == NULL) {
            addTerm(&result, poly2->coefficient, poly2->exponent);
            poly2 = poly2->next;
        } else if (poly2 == NULL) {
            addTerm(&result, poly1->coefficient, poly1->exponent);
            poly1 = poly1->next;
        } else if (poly1->exponent > poly2->exponent) {
            addTerm(&result, poly1->coefficient, poly1->exponent);
            poly1 = poly1->next;
        } else if (poly1->exponent < poly2->exponent) {
            addTerm(&result, poly2->coefficient, poly2->exponent);
            poly2 = poly2->next;
        } else {
            addTerm(&result, poly1->coefficient + poly2->coefficient, poly1-
>exponent);
            poly1 = poly1->next;
            poly2 = poly2->next;
        }
    }
```

```
    return result;
}

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

    readPolynomial(&poly1);

    readPolynomial(&poly2);

    printPolynomial(poly1);

    printPolynomial(poly2);

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

    printPolynomial(result);

    return 0;
}
```

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

3. 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 25
typedef struct {
```

```
    int co;int ex;
}term;
void multi(term poly1[],int n,term poly2[],int m,term r[],int *size)
{
    int i,j,k;
    *size=0;
    for( i=0;i<n;i++)
    {
        for( j=0;j<m;j++)
        {
            int newco=poly1[i].co*poly2[j].co;
            int newexp=poly1[i].ex+poly2[j].ex;
            int f=0;
            for(k=0;k<*size;k++)
            {
                if(r[k].ex==newexp)
                {
                    r[k].co+=newco;
                    f=1;
                    break;
                }
            }
            if(!f)
            {
                r[*size].co=newco;
                r[*size].ex=newexp;
                (*size)++;
            }
        }
    }
}
void del(term r[],int *size,int ext){
    int i=0;
 while(i<*size)
    {
        if(r[i].ex==ext){
            for(int j=i;j<(*size)-1;j++){
                r[j]=r[j+1];
            }
            (*size)--;
        }else{i++;
        }
```

```
    }
  }


void print(term poly[],int size)
{
  for(int i=0;i<size;i++)
  {
    printf("%d",poly[i].co);
    if(poly[i].ex==1)
    {
      printf("x");

    }else if(poly[i].ex>1)
    {
       printf("x^%d",poly[i].ex);
    }
    if(i<size-1)
    {
      printf(" + ");
    }
  }
  printf("\n");
}
int main()
{
  int n,m ,ext;
  term poly1[5],poly2[5],r[MAX];
  scanf("%d",&n);
  for(int i=0;i<n;i++)
  {
    scanf("%d %d ",&poly1[i].co,&poly1[i].ex);
  }
   scanf("%d",&m);
  for(int i=0;i<m;i++)
  {
    scanf("%d %d ",&poly2[i].co,&poly2[i].ex);
  }
  scanf("%d",&ext);
```

```
    int rs;
    multi(poly1,n,poly2,m,r,&rs);
    printf("Result of the multiplication: ");
    print(r,rs);
    del(r,&rs,ext);
    printf("Result after deleting the term: ");
    print(r,rs);
    return 0;

}
```

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