Rajalakshmi Engineering College

Name: Sajine Santhakumar
Email: 240701459@rajalakshmi.edu.in
Roll no: 240701459
Phone: 9952076750
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_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 25

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

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
    int coe;
    int expo;
```

```
    struct node* next;
};Node;
Node* createNode(int coe,int expo){
    Node* newNode=(Node*)malloc(sizeof(Node));
    newNode->coe=coe;
    newNode->expo=expo;
    newNode->next=NULL;
    return newNode;
}
void insertTerm(Node** head,int coe,int expo){
    if(coe==0){
        return;
    }
    Node* newNode=createNode(coe,expo);
    if(*head==NULL||expo<(*head)->expo){
        newNode->next=*head;
        *head=newNode;
        return;
    }
    Node* current=*head;
    while(current->next!=NULL && current->next->expo<expo){
        current=current->next;
    }
    if(current->next!=NULL && current->next->expo==expo){
        current->next->coe+=coe;
        if(current->next->coe==0){
            Node* temp=current->next;
            current->next=current->next->next;
            free(temp);
        }
        free(newNode);
    }else{
        newNode->next=current->next;
        current->next=newNode;
    }
}
Node* addPoly(Node* poly1,Node* poly2){
    Node* result=NULL;
    while(poly1!=NULL||poly2!=NULL){
        int coe,expo;
        if(poly1==NULL){
            coe=poly2->coe;
```

```
            expo=poly2->expo;
            poly2=poly2->next;
        }
        else if(poly2==NULL){
            coe=poly1->coe;
            expo=poly1->expo;
            poly1=poly1->next;
        }
        else if(poly1->expo<poly2->expo){
            coe=poly1->coe;
            expo=poly1->expo;
            poly1=poly1->next;
        }
        else if(poly1->expo>poly2->expo){
            coe=poly2->coe;
            expo=poly2->expo;
            poly2=poly2->next;
        }
        else{
            coe=poly1->coe+poly2->coe;
            expo=poly1->expo;
            poly1=poly1->next;
            poly2=poly2->next;
        }
        insertTerm(&result,coe,expo);
    }
    return result;
}
void printpoly(Node* head){
    if(head==NULL){
        printf("0\n");
        return;
    }
    Node* current=head;
    while(current!=NULL){
        printf("%dx^%d ",current->coe,current->expo);
        if(current->next!=NULL){
            printf(" + ");
        }
        current=current->next;
    }
    printf("\n");
```

```
}
void freelist(Node* head){
    while(head!=NULL){
        Node* temp=head;
        head=head->next;
        free(temp);
    }
}
int main(){
    Node* poly1=NULL;
    Node* poly2=NULL;
    int coe,expo;
    while(1){
        scanf("%d %d",&coe,&expo);
        if(coe==0 && expo==0){
            break;
        }
        insertTerm(&poly1,coe,expo);
    }
    while(1){
        scanf("%d %d",&coe,&expo);
        if(coe==0 && expo==0){
            break;
        }
        insertTerm(&poly2,coe,expo);
    }
    printpoly(poly1);
    printpoly(poly2);
    Node* result=addPoly(poly1,poly2);
    printpoly(result);
    freelist(poly1);
    freelist(poly2);
    freelist(result);
    return 0;
}
```

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

2. 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>
struct poly{
   int coeff;
   int expo;
   struct poly *Next;
};
typedef struct poly Poly;
void create(Poly *);
void multiplication(Poly *,Poly *,Poly *);
void deletemid(Poly *,int );
void display(Poly *);

int main(){
   int dex,n,m;
   Poly *poly1=(Poly*)malloc(sizeof(Poly));
   Poly *poly2=(Poly*)malloc(sizeof(Poly));
   Poly *result=(Poly*)malloc(sizeof(Poly));
   poly1->Next=NULL;
   poly2->Next=NULL;
   if(scanf("%d",&n)!=1) return 1;
   for(int i=0;i<n;i++){
      create(poly1);
   }
   if(scanf("%d",&m)!=1) return 1;
   for(int i=0;i<m;i++){
      create(poly2);
   }
   multiplication(poly1,poly2,result);
   printf("Result of the multiplication:");
   display(result);
   if(scanf("%d",&dex)!=1) return 1;
   deletemid(result,dex);
```

```
    printf("Result after deleting the term:");
    display(result);
    return 0;
}

void create(Poly * List){
    Poly *position,*newnode;
    position=List;
    newnode=(Poly*)malloc(sizeof(Poly));
    if(scanf("%d %d",&newnode->coeff,&newnode->expo)!=2){
        free(newnode);
        return;
    }
    newnode->Next=NULL;
    while(position->Next!=NULL){
        position=position->Next;
    }
    position->Next=newnode;
}

void display(Poly *List){
    Poly *position;
    position=List->Next;
    while(position!=NULL){
        if(position->expo!=1){
            printf(" %dx^%d ",position->coeff,position->expo);
        }
        else{
            printf(" %dx ",position->coeff);
        }
        position=position->Next;
        if(position!=NULL && position->coeff>0){
            printf("+");
        }
    }
    printf("\n");
}

void deletemid(Poly *List,int dex){
    Poly *position,*TempNode;
    position=List;
    while(position->Next!=NULL && position->Next->expo!=dex){
```

```
        position=position->Next;
    }
    if(!(position->Next==NULL)){
        TempNode=position->Next;
        position->Next=TempNode->Next;
        free(TempNode);
    }
}

void multiplication(Poly *poly1,Poly *poly2,Poly *result){
    if(!poly1->Next || !poly2->Next) return;

    for(Poly *ptr1=poly1->Next;ptr1;ptr1=ptr1->Next){
        for(Poly *ptr2=poly2->Next;ptr2;ptr2=ptr2->Next){
            int newcoeff=ptr1->coeff*ptr2->coeff;
            int newexpo=ptr1->expo+ptr2->expo;

            poly *respos=result;
            while(respos->Next && respos->Next->expo>newexpo){
                respos=respos->Next;
            }
            if(respos->Next && respos->Next->expo==newexpo){
                respos->Next->coeff+=newcoeff;
            }
            else{
                Poly *newnode=(Poly*)malloc(sizeof(Poly));
                newnode->coeff=newcoeff;
                newnode->expo=newexpo;
                newnode->Next=respos->Next;
                respos->Next=newnode;
            }
        }
    }
}
```

Status : Partially correct **Marks : 5/10**

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

```
// You are using GCC
#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 insert(Node**poly,int coeff,int exp)
{
    Node*newNode=createNode(coeff,exp);
    if(*poly==NULL)
    {
        *poly=newNode;
    }else{
        Node*temp=*poly;
        while(temp->next!=NULL)
            temp=temp->next;
        temp->next=newNode;
        }
    }
    Node*multiplyPolynomials(Node*poly1,Node*poly2){
        if(!poly1||!poly2)return NULL;

        Node*result=NULL;
        Node*temp1=poly1;
        Node*temp2;

        while(temp1){
            temp2=poly2;
            while(temp2){
                int coeff=temp1->coeff*temp2->coeff;
                int exp=temp1->exp+temp2->exp;
                insert(&result,coeff,exp);
```

```
        temp2=temp2->next;
    }
    temp1=temp1->next;
}

Node*ptr1,*ptr2,*prev;
ptr1=result;
while(ptr1 && ptr1->next){
    prev=ptr1;
    ptr2=ptr1->next;
    while(ptr2){
        if(ptr1->exp==ptr2->exp){
            ptr1->coeff+=ptr2->coeff;
            prev->next=ptr2->next;
            free(ptr2);
            ptr2=prev->next;
        }else{
            prev=ptr2;
            ptr2=ptr2->next;
        }
    }
    ptr1=ptr1->next;
}
    return result;
}
void printPolynomial(Node*poly){
    int first=1;
    while(poly){
        if(!first)printf(" + ");
        if(poly->exp==0){
            printf("%d",poly->coeff);
        }else if(poly->exp==1){
        printf("%dx",poly->coeff);
        }else {
            printf("%dx^%d",poly->coeff,poly->exp);
        }
        first=0;
        poly=poly->next;
    }
    printf("\n");
}
```

```
Node*readPolynomials(){
    Node*poly=NULL;
    int coeff,exp;
    char choice;

    do{
        scanf("%d",&coeff);
        scanf("%d",&exp);
        insert(&poly,coeff,exp);
        getchar();

        scanf("%c",&choice);
    }while(choice=='y'||choice=='Y');

    return poly;
}
int main(){
    Node*poly1=readPolynomials();
    Node*poly2=readPolynomials();

    Node* result=multiplyPolynomials(poly1,poly2);
    printPolynomial(result);
    return 0;
}
```

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