Rajalakshmi Engineering College

Name: Sarvesh S
Email: 240701479@rajalakshmi.edu.in
Roll no: 240701479
Phone: 9361488694
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

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***

2. Problem Statement

John is working on a math processing application, and his task is to simplify polynomials entered by users. The polynomial is represented as a linked list, where each node contains two properties:

Coefficient of the term.

Exponent of the term.

John's goal is to combine all the terms that have the same exponent, effectively simplifying the polynomial.

Input Format

The first line of input consists of an integer representing the number of terms in the polynomial.

The next n lines of input consist of two integers, representing the coefficient and exponent of the polynomial in each line separated by space.

Output Format

The first line of output prints the original polynomial in the format 'cx^e + cx^e + ...' (where c is the coefficient and e is the exponent of each term).

The second line of output displays the simplified polynomial in the same format as the original polynomial.

If the polynomial is 0, then only '0' will be printed.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
5 2
3 1
6 2
Output: Original polynomial: 5x^2 + 3x^1 + 6x^2
Simplified polynomial: 11x^2 + 3x^1

Answer

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int coeff;
   int expo;
   struct node *Next;
};
typedef struct node Poly;
void create(Poly *);
void display(Poly *);
void simplify(Poly *poly1,Poly *result);

int main(){
   int n;
   Poly *poly1=(Poly*)malloc(sizeof(Poly));
```

```
    Poly *result=(Poly*)malloc(sizeof(Poly));
    poly1->Next=NULL;
    if(scanf("%d",&n)!=1) return 1;
    for(int i=0;i<n;i++){
        create(poly1);
    }
    printf("Original polynomial:");
    display(poly1);
    simplify(poly1,result);
    printf("Simplified polynomial:");
    display(result);
    return 0;
}

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

void simplify(Poly *poly1,Poly *result){
    Poly *pos1=poly1->Next;
    Poly *temp,*rpos=result;
    while(pos1!=NULL){
        temp=result->Next;
        while(temp!=NULL){
            if(temp->expo==pos1->expo){
                temp->coeff+=pos1->coeff;
                break;
            }
            temp=temp->Next;
        }
        if(temp==NULL){
            Poly *newnode=(Poly*)malloc(sizeof(Poly));
            newnode->coeff=pos1->coeff;
            newnode->expo=pos1->expo;
            newnode->Next=NULL;
            rpos->Next=newnode;
            rpos=newnode;
        }
        pos1=pos1->Next;
    }
}

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;
}
```

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

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

```
#include<stdio.h>
#include<stdlib.h>
struct poly{
    int coeff;
    int expo;
    struct poly *Next;
};
typedef struct poly Poly;
void create(Poly *);
void display(Poly *poly1,Poly *poly2);
void cequality(Poly *poly1,Poly *poly2);
int main(){
    int n,m;
    Poly *poly1=(Poly*)malloc(sizeof(Poly));
    Poly *poly2=(Poly*)malloc(sizeof(Poly));
    poly1->Next=NULL;
    poly1->Next=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        create(poly1);
    }
    scanf("%d",&m);
    for(int i=0;i<m;i++){
        create(poly2);
    }
    display(poly1,poly2);
    if(n!=m){
        printf("Polynomials are Not Equal.");
        return 0;
    }
    else{
        cequality(poly1,poly2);
    }
    free(poly1);
    free(poly2);
    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 *poly1,Poly *poly2){
    Poly *pos1,*pos2;
    pos1=poly1->Next;
    pos2=poly2->Next;
    printf("Polynomial 1:");
    while(pos1!=NULL){
        printf(" (%dx^%d) ",pos1->coeff,pos1->expo);
        pos1=pos1->Next;
        if(pos1!=NULL && (pos1->coeff>0 || pos1->coeff<=0)){
            printf("+");
        }
    }
    printf("\n");
    printf("Polynomial 2:");
    while(pos2!=NULL){
        printf(" (%dx^%d) ",pos2->coeff,pos2->expo);
        pos2=pos2->Next;
        if(pos2!=NULL && (pos2->coeff>0 || pos2->coeff<=0)){
            printf("+");
        }
    }
    printf("\n");
}

void cequality(Poly *poly1,Poly *poly2){
    Poly *pos1,*pos2;
    pos1=poly1->Next;
    pos2=poly2->Next;
    while(pos1!=NULL && pos2!=NULL){
        if(pos1->coeff!=pos2->coeff || pos1->expo != pos2->expo){
            printf("Polynomials are Not Equal.\n");
            return;
        }
        pos1=pos1->Next;
        pos2=pos2->Next;
    }
    printf("Polynomials are Equal.\n");
}
```

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