Rajalakshmi Engineering College

Name: shiloh .s
Email: 240701498@rajalakshmi.edu.in
Roll no: 240701498
Phone: 9488883273
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

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;
};
struct node*createnode(int coeff,int expo)
{
   struct node*newnode=(struct node*)malloc(sizeof(struct node));
   newnode->coeff=coeff;
   newnode->expo=expo;
   newnode->next=NULL;
   return newnode;
}
void insertterm(struct node**head,int coeff,int expo)
```

```
{
   struct node*newnode=createnode(coeff,expo);
   if(*head==NULL)
   {
      *head=newnode;
   }else
   {
      struct node*temp=*head;
      while(temp->next!=NULL)
      {
         temp=temp->next;
      }
      temp->next=newnode;
   }
}
void simplifypolynomial(struct node*head)
{
   struct node*current,*pre,*temp;
   current=head;
   while(current!=NULL&&current->next!=NULL)
   {
      pre=current;
      temp=current->next;
      while(temp!=NULL)
      {
         if(current->expo==temp->expo)
         {
            current->coeff+=temp->coeff;
            pre->next=temp->next;
            free(temp);
            temp=pre->next;
         }
         else{
            pre=temp;
            temp=temp->next;
         }
      }
      current=current->next;
   }
}
void printpolynomial(struct node*head)
{
```

```
    struct node*temp=head;
    int first=1;
    while(temp!=NULL)
    {
          if(!first)
          {
             printf("+ ");
          }
          printf("%dx^%d ",temp->coeff,temp->expo);
          first=0;
       temp=temp->next;
    }
    printf("\n");
}
int main(){
    struct node*polynomial=NULL;
    int numterms,coeff,expo;
    scanf("%d",&numterms);
    for(int i=0;i<numterms;i++)
    {
       scanf("%d %d",&coeff,&expo);
       insertterm(&polynomial,coeff,expo);
    }
    printf("Original polynomial:");
    printpolynomial(polynomial);
    simplifypolynomial(polynomial);
    printf("Simplified polynomial:");
    printpolynomial(polynomial);
    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 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 addterm(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->exponent==exponent){
         temp->coefficient+=coefficient;
         free(newterm);
      }else{
         newterm->next=temp->next;
         temp->next=newterm;
      }
   }
}
void multiplypolynomials(struct term*poly1,struct term*poly2,struct
term**result)
```

```
{
    struct term*temp1=poly1;
    while(temp1!=NULL)
    {
        struct term*temp2=poly2;
        while(temp2!=NULL)
        {
            int newcoeff=temp1->coefficient*temp2->coefficient;
            int newexp=temp1->exponent+temp2->exponent;
            addterm(result,newcoeff,newexp);
            temp2=temp2->next;
        }
        temp1=temp1->next;
    }
}
void deleteterm(struct term**poly,int exponenttodelete){
    struct term*temp=*poly,*prev=NULL;
    while(temp!=NULL){
        if(temp->exponent==exponenttodelete){
            if(prev==NULL){
                *poly=temp->next;
            }else{
                prev->next=temp->next;
            }
            free(temp);
            return;
        }
        prev=temp;
        temp=temp->next;
    }
}
void printpolynomial(struct term*poly){
    if(poly==NULL)
    {
        printf("0\n");
        return;
    }
    struct term*temp=poly;
    while(temp!=NULL){
        if(temp->coefficient>0&&temp!=poly){
            printf(" + ");
        }
```

```
        if(temp->exponent!=1)
        {
        printf("%dx^%d",temp->coefficient,temp->exponent);
        }
        else
        {
            printf("%dx",temp->coefficient);
        }
        temp=temp->next;
    }
    printf("\n");
}
int main()
{
    int n1,n2,exponenttodelete;
    scanf("%d",&n1);
    struct term*poly1=NULL;
    for(int i=0;i<n1;i++){
        int coeff,expe;
        scanf("%d %d",&coeff,&expe);
        addterm(&poly1,coeff,expe);
    }
    scanf("%d",&n2);
    struct term*poly2=NULL;
    for(int i=0;i<n2;i++){
        int coeff,expe;
        scanf("%d %d",&coeff,&expe);
        addterm(&poly2,coeff,expe);
    }
    scanf("%d",&exponenttodelete);
    struct term*result=NULL;
    multiplypolynomials(poly1,poly2,&result);
    printf("Result of the multiplication: ");
    printpolynomial(result);
    deleteterm(&result,exponenttodelete);
    printf("Result after deleting the term: ");
    printpolynomial(result);
    return 0;
}
```

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

3. 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>
struct poly{
   int coeff;
   int expo;
   struct poly*next;
};
void create(struct poly*poly)
{
   int a;
   struct poly*position=poly;
   scanf("%d",&a);
   for(int i=0;i<a;i++)
   {
      struct poly*newnode=(struct poly*)malloc(sizeof(struct poly));
      newnode->next=NULL;
      scanf("%d %d",&newnode->coeff,&newnode->expo);
      position->next=newnode;
      position=newnode;
   }
}
void display(struct poly*poly)
{
   struct poly*temp=poly->next;
   if(temp==NULL)
   {
      printf("0\n");
   }
   while(temp!=NULL)
   {  if(temp->coeff>0&&temp!=poly->next)
      {
         printf(" + ");
      }
      if(temp->expo==0)
```

```
        {
            printf("%d",temp->coeff);
        }
        else
        if(temp->expo==1)
        {
            printf("%dx",temp->coeff);
        }
        else
        {
             printf("%dx^%d",temp->coeff,temp->expo);
        }
        temp=temp->next;
    }
}
int main()
{
    struct poly*poly1=(struct poly*)malloc(sizeof(struct poly));
    struct poly*poly2=(struct poly*)malloc(sizeof(struct poly));
    poly1->next=NULL;
    poly2->next=NULL;
    create(poly1);
    create(poly2);
    display(poly1);
    printf("\n");
    display(poly2);
    printf("\n");
    return 0;
}
```

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