Rajalakshmi Engineering College

Name: SHREYA KS
Email: 240701506@rajalakshmi.edu.in
Roll no: 240701506
Phone: 9789293683
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 : 20

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

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* 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**head,int coeff,int exp)
{
   Node*newNode=createNode(coeff,exp);
   if(*head==NULL||(*head)->exp>exp)
   {
      newNode->next=*head;
      *head=newNode;
      return;
   }
   Node*temp=*head;
   while(temp->next!=NULL&&temp->next->exp<exp)
   {
      temp=temp->next;
   }
   if(temp->next!=NULL&&temp->next->exp==exp)
   {
      temp->next->coeff+=coeff;
      free(newNode);
      if(temp->next->coeff==0){
         Node*toDelete =temp->next;
         temp->next=temp->next;
```

```
            free(toDelete);
        }
        }else
        {
            newNode->next=temp->next;
            temp->next=newNode;
        }
}

void display(Node*head)
{
    if(head==NULL)
    {
        printf("0\n");
        return;
    }
    while(head!=NULL)
    {
        printf("%dx^%d",head->coeff,head->exp);
        if(head->next!=NULL)
        printf(" + ");
        head=head->next;
    }
    printf("\n");
}
Node*addPolynomials(Node*p1,Node*p2)
{
    Node*result=NULL;
    while(p1!=NULL||p2!=NULL){
        if(p1==NULL){
            insert(&result,p2->coeff,p2->exp);
            p2=p2->next;
        }else if(p2==NULL){
            insert(&result,p1->coeff,p1->exp);
            p1=p1->next;
        }else if(p1->exp == p2->exp){
            int sum = p1->coeff+p2->coeff;
            if(sum!=0){
                insert(&result,sum,p1->exp);
            }
            p1=p1->next;
            p2=p2->next;
        }else if(p1->exp < p2->exp){
            insert(&result,p1->coeff,p1->exp);
        p1=p1->next;
        }
        else{
            insert(&result,p2->coeff,p2->exp);
            p2=p2->next;
        }
    }
    return result;
}
Node*readPolynomial()
{
    Node*poly=NULL;
    int coeff,exp;
    while(1)
    {
        scanf("%d %d",&coeff,&exp);
        if(coeff==0 && exp==0)
        break;
        insert(&poly,coeff,exp);
    }
    return poly;
}
int main()
{
    Node*p1,*p2,*result;
    p1=readPolynomial();
    p2=readPolynomial();

    display(p1);
    display(p2);
    result=addPolynomials(p1,p2);
    display(result);
    return 0;
}
```

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

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

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

typedef struct Node{
   int coeff;
   int exp;
   struct Term* next;
}Node;

struct Term* createNode(int coeff,int exp) {
   Node* newNode=(Node*)malloc(sizeof(Node));
   newNode->coeff=coeff;
   newNode->exp=exp;
   newNode->next=NULL;
   return newNode;
}

void insertterm(Node** poly,int coeff,int exp) {
   Node* newNode=createNode(coeff,exp);

}

Node* multiplypolynomials(Node* poly1,Node* poly2){
   Node*result=NULL;
   for(Node*p1=poly1;p1!=NULL;p1=p1->next) {
      int newcoeff=p1->coeff*p1->coeff;
      int newexp=p1->exp+p1->exp;

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

   }

return result;
}

void printpolynomial(Node*poly){
   while(poly!=NULL){
      printf("%dx^%d",poly->coeff,poly->exp);
      if(poly->next !=NULL) printf(" + ");
      poly=poly->next;
   }
   printf("\n");
}
int main(){
   Node* poly1=NULL;
   Node* poly2=NULL;

   insertterm(&poly1, 3, 1);
   insertterm(&poly1, 10, 0);
   insertterm(&poly2, 2, 2);
   insertterm(&poly2, 3, 1);
   insertterm(&poly2, 2, 0);

   printf("Polynomial 1: ");
   printpolynomial(poly1);
   printf("Polynomial 2: ");
   printpolynomial(poly2);

   Node* result=multiplypolynomials(poly1,poly2);
   printf("Resultant Polynomial: ");
   printpolynomial(result);

   return 0;
}
```

Status : Wrong ***Marks : 0/10***