Rajalakshmi Engineering College

Name: Santhosh G
Email: 240701473@rajalakshmi.edu.in
Roll no: 240701473
Phone: 8883772237
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 : 30

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

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct node {
   int coeff;
   int exp;
   struct node* next;
}Node;
Node* create(int coeff, int exp){
   Node* newNode=(Node*)malloc(sizeof(Node));
   newNode->coeff=coeff;
   newNode->exp=exp;
   newNode->next=0;
   return newNode;
}
void insert(Node** head,int coeff,int exp){
   Node* newNode=create(coeff,exp);
   if(*head==0){
       *head=newNode;
   }
   else{
       node* temp=*head;
       while(temp->next!=0){
           temp=temp->next;
       }
       temp->next=newNode;
   }
}
void printList(Node* head){
   if(!head){
       printf("0x^0\n");
   }
   node*temp=head;
   while(temp){
       printf("%dx^%d",temp->coeff,temp->exp);
       if(temp->next)
       printf(" + ");
       temp=temp->next;
   }
   printf("\n");
}
Node* simplify(Node* head){
   if(!head){
       return 0;
   }
   Node* result=0;
   Node* temp=head;
   while(temp){
       Node* search=result;
       int found=0;
       while(search){
           if(search->exp==temp->exp){
               search->coeff+=temp->coeff;
               found=1;
               break;
           }
           search=search->next;
       }
       if(!found){
           insert(&result,temp->coeff,temp->exp);
       }
       temp=temp->next;
   }
   Node* prev=0;
   Node* current=result;
   while(current){
       if(current->coeff==0){
           if(prev){
               prev->next=current->next;
           }
           else{
               result=current->next;
               node*total=current;
               current=current->next;
               free(total);
           }
       }
       else{
           prev=current;
           current=current->next;
       }
   }
   return result;
}
void freeList(Node* head){
   Node* temp;
   while(head){
       temp=head;
       head=head->next;
       free(temp);
   }
}
int main()
{
   int n,coeff,exp;
   Node* head=0;
   scanf("%d",&n);
   for(int i=0;i<n;i++){
       scanf("%d %d",&coeff,&exp);
       insert(&head,coeff,exp);
   }
   printf("Original polynomial: ");
   printList(head);
   Node* simplified=simplify(head);
   printf("Simplified polynomial: ");
   printList(simplified);
   freeList(head);
   freeList(simplified);
   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

```
// 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 insertTerm(Node** head,int coeff,int exp){
   if(coeff==0){
       return;
   }
   Node* newNode=createNode(coeff,exp);
   if(*head==NULL||exp<(*head)->exp){
       newNode->next=*head;
       *head=newNode;
       return;
   }
   Node* current=*head;
   while(current->next!=NULL && current->next->exp<exp){
       current=current->next;
   }
   if(current->next!=NULL && current->next->exp==exp){
       current->next->coeff+=coeff;
       if(current->next->coeff==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 coeff,exp;
       if(poly1==NULL){
           coeff=poly2->coeff;
           exp=poly2->exp;
           poly2=poly2->next;
       }
       else if(poly2==NULL){
           coeff=poly1->coeff;
           exp=poly1->exp;
           poly1=poly1->next;
       }
       else if(poly1->exp<poly2->exp){
           coeff=poly1->coeff;
           exp=poly1->exp;
           poly1=poly1->next;
       }
       else if(poly1->exp>poly2->exp){
           coeff=poly2->coeff;
           exp=poly2->exp;
           poly2=poly2->next;
       }
       else{
           coeff=poly1->coeff+poly2->coeff;
           exp=poly1->exp;
           poly1=poly1->next;
           poly2=poly2->next;
       }
       insertTerm(&result,coeff,exp);
   }
   return result;
}
void printpoly(Node* head){
   if(head==NULL){
       printf("0\n");
       return;
   }
   Node* current=head;
   while(current!=NULL){
       printf("%dx^%d",current->coeff,current->exp);
       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 coeff,exp;
  while(1){
    scanf("%d %d",&coeff,&exp);
    if(coeff==0 && exp==0){
      break;
    }
    insertTerm(&poly1,coeff,exp);
  }
  while(1){
    scanf("%d %d",&coeff,&exp);
    if(coeff==0&&exp==0){
      break;
    }
    insertTerm(&poly2,coeff,exp);
  }
  printpoly(poly1);
  printpoly(poly2);
  Node* result=addPoly(poly1,poly2);
  printpoly(result);
  freeList(poly1);
  freeList(poly2);
  freeList(result);
  return 0;
}
```

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

```
// 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** head, int coeff, int exp){
   Node* newNode=createNode(coeff,exp);
   if(exp<0){
      free(newNode);
      return;
   }
   if(*head==NULL||exp>(*head)->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);
   }
   else{
      newNode->next=temp->next;
      temp->next=newNode;
   }
}
void printList(Node* head){
   if(head==NULL){
        printf("0\n");
        return;
    }
    Node*temp=head;
    while(temp!=NULL){
        printf("(%dx^%d)",temp->coeff,temp->exp);
        if(temp->next) printf(" + ");
        temp=temp->next;
    }
    printf("\n");
}
int compare(Node* poly1,Node* poly2){
    while(poly1&&poly2){
        if(poly1->coeff!=poly2->coeff||poly1->exp!=poly2->exp)
            return 0;
        poly1=poly1->next;
        poly2=poly2->next;
    }
    return (poly1==NULL&&poly2==NULL);
}
void freeList(Node*head){
    while(head!=NULL){
        Node* temp=head;
        head=head->next;
        free(temp);
    }
}
int main(){
    Node* poly1=NULL;
    Node* poly2=NULL;
    int n,coeff,exp;
    scanf("%d",&n);
    while(n--){
        scanf("%d %d",&coeff,&exp);
        insert(&poly1,coeff,exp);
    }
    scanf("%d",&n);
    while(n--){
        scanf("%d %d",&coeff,&exp);
        insert(&poly2,coeff,exp);
    }
    printf("Polynomial 1: ");
    printList(poly1);
    printf("Polynomial 2: ");
    printList(poly2);
    if(compare(poly1,poly2)){
        printf("Polynomials are Equal.\n");
    }
    else{
        printf("Polynomials are Not Equal.\n");
    }
    freeList(poly1);
    freeList(poly2);
    return 0;
}
```

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