Rajalakshmi Engineering College

Name: NEIL DANIEL A
Email: 240701356@rajalakshmi.edu.in
Roll no: 240701356
Phone: 8925059757
Branch: REC
Department: I CSE FD
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

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node{
   int coe;
   int expo;
   struct node* next;
}Node;
Node* create(int coe,int expo){
   Node* newNode=(Node*)malloc(sizeof(Node));
   newNode->coe=coe;
   newNode->expo=expo;
   newNode->next=0;
   return newNode;
}
void insert(Node**head,int coe,int expo){
   Node* newNode=create(coe,expo);
```

```
    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");
        return;
    }
    node* temp=head;
    while(temp){
        printf("%dx^%d",temp->coe,temp->expo);
        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->expo==temp->expo){
                search->coe+=temp->coe;
                found=1;
                break;
            }
            search= search->next;
        }
        if(!found){
            insert(&result,temp->coe,temp->expo);
        }
        temp=temp->next;
    }
    Node* prev=0;
    Node* current=result;
    while(current){
        if(current->coe==0){
            if(prev){
                prev->next=current->next;
            }
            else{
                result=current->next;
                node*todel=current;
                current=current->next;
                free(todel);
            }
        }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,coe,expo;
    Node* head=0;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d %d",&coe,&expo);
        insert(&head,coe,expo);
    }
    printf("Original polynomial: ");
    printList(head);
    Node* simplified=simplify(head);
    printf("Simplified polynomial: ");
    printList(simplified);
    freelist(head);
    freelist(simplified);

}
```

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

2. 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>
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 insert(Node** head,int coe,int expo){
   Node* newNode=createNode(coe,expo);
   if(expo<0){
      free(newNode);
      return;
   }
   if(*head==NULL||expo>(*head)->expo){
      newNode->next=*head;
      *head=newNode;
```

```
        return;
    }
    Node* temp=*head;
    while(temp->next!=NULL && temp->next->expo>expo){
        temp=temp->next;
    }
    if(temp->next!=NULL && temp->next->expo==expo){
        temp->next->coe=coe;
        free(newNode);
    }
    else{
        newNode->next=temp->next;
        temp->next=newNode;
    }
}
void printList(Node* head){
    if(head==NULL){
        printf("0\n");
        return;
    }
    Node*temp=head;
    int first = 1;
    while(temp!=NULL){
        if(temp->coe!=0){
            if(!first && temp->coe>0){
                printf(" + ");
            }
            if(temp->expo==1){
                printf("%dx",temp->coe);
            }
            else if(temp->expo==0){
                printf("%d",temp->coe);
            }
            else if(temp->expo<0){
                continue;
            }
            else{
                printf("%dx^%d",temp->coe,temp->expo);
            }
            first=0;
        }
        temp=temp->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 n,coe,expo;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d %d",&coe,&expo);
        insert(&poly1,coe,expo);
    }
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d %d",&coe,&expo);
        insert(&poly2,coe,expo);
    }
    printList(poly1);
    printList(poly2);
    freeList(poly1);
    freeList(poly2);
}
```

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

3. 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 <stdlib.h>
#include <stdio.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***