Rajalakshmi Engineering College

Name: ROHITH KUMAR S
Email: 240701436@rajalakshmi.edu.in
Roll no: 240701436
Phone: 7603815548
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 : 2
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. 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 c;
   int e;
   struct node *next;
}Node;
Node *createNode(int c,int e){
   Node *newnode=(Node*)malloc(sizeof(Node));
```

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

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

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>
typedef struct node{
   int c;
   int e;
   struct node *next;
}Node;
Node *create(int c,int e){
   Node *newnode=(Node*)malloc(sizeof(Node));
   newnode->c=c;
   newnode->e=e;
   newnode->next=0;
   return newnode;
}
void insert(Node **head,int c,int e){
   Node *newnode=create(c,e);
   if(*head==0){
      *head=newnode;
   }else{
      Node *temp=*head;
      while(temp->next!=0){
         temp=temp->next;
      }
      temp->next=newnode;
   }
}
void print(Node *head){
   if(!head){
      printf("0x^0\n");
      return;
   }
   Node *temp=head;
   while(temp){
      printf("%dx^%d",temp->c,temp->e);
      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->e==temp->e){
                search->c+=temp->c;
                found=1;
                break;
            }
            search=search->next;
        }
        if(!found){
            insert(&result,temp->c,temp->e);
        }
        temp=temp->next;
    }
    Node *prev=0;
    Node *current=result;
    while(current){
        if(current->c==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 fre(Node *head){
   Node *temp;
   while(head){
      temp=head;
      head=head->next;
      free(temp);
   }
}
int main(){
   int n,c,e;
   Node *head=0;
   scanf("%d",&n);
   for(int i=0;i<n;i++){
      scanf("%d %d",&c,&e);
      insert(&head,c,e);
   }
   printf("Original polynomial: ");
   print(head);
   Node *simplified=simplify(head);
   printf("Simplified polynomial: ");
   print(simplified);
   fre(head);
   fre(simplified);
}
```

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 node{
   int c;
   int e;
   struct node *next;
};
typedef struct node Node;
int main(){
   Node *List=NULL;
   Node *pos,*newnode;
   int n;
   scanf("%d",&n);
   for(int i=0;i<n;i++){
      newnode=(Node*)malloc(sizeof(Node));
      scanf("%d %d",&newnode->c,&newnode->e);
      if(List==NULL){
         List=newnode;
         pos=newnode;
      }else{
         pos->next=newnode;
         pos=newnode;
      }
   }
   Node *List1=NULL;
   Node *pos1,*newnode1;
   int m;
   scanf("%d",&m);
   for(int i=0;i<m;i++){
      newnode1=(Node*)malloc(sizeof(Node));
      scanf("%d %d",&newnode1->c,&newnode1->e);
      if(List1==NULL){
         List1=newnode1;
         pos1=newnode1;
      }else{
         pos1->next=newnode1;
         pos1=newnode1;
      }
   }
   pos=List;
   printf("Polynomial 1: ");
   while(pos!=NULL){
      printf("(%dx^%d)",pos->c,pos->e);
```

```
        pos=pos->next;
        if(pos!=NULL){
            printf(" + ");
        }
    }
    printf("\n");
     pos1=List1;
    printf("Polynomial 2: ");
    while(pos1!=NULL){
        printf("(%dx^%d)",pos1->c,pos1->e);
        pos1=pos1->next;
        if(pos1!=NULL){
            printf(" + ");
        }
    }
    printf("\n");
    if(m!=n){
        printf("Polynomials are Not Equal.");
    }else{
        int v=0;
        pos=List;
        pos1=List1;
        while(pos!=NULL&&pos1!=NULL){
            if(pos->e!=pos1->e || pos->c!=pos1->c){
                v=1;
                break;
            }
            pos=pos->next;
            pos1=pos1->next;
        }
        if(v==1){
            printf("Polynomials are Not Equal.");
        }else{
            printf("Polynomials are Equal.");
        }
    }
}
```

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