# Rajalakshmi Engineering College

Name: Nikhilesh D
Email: 240701360@rajalakshmi.edu.in
Roll no: 240701360
Phone: 9940617256
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 : 20

## Section 1 : Coding

1. 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;
}

Node* insert(Node* head,int coeff,int exp){
    Node* newNode=createNode(coeff,exp);
    if(!head) return newNode;
    Node* temp=head;
    while(temp->next)
        temp=temp->next;
    temp->next=newNode;
    return head;
}

void displayPoly(Node* head){
    while(head){
        printf("(%dx^%d)",head->coeff,head->exp);
        if(head->next) printf(" + ");
        head=head->next;
    }
    printf("\n");
}

int areEqual(Node* p1,Node* p2){
    while(p1 && p2){
        if(p1->coeff!= p2->coeff || p1->exp!=p2->exp)
            return 0;
        p1=p1->next;
        p2=p2->next;
    }
    return (p1==NULL && p2==NULL);
```

```
}
int main(){
    int n,m,coeff,exp;
    Node *poly1=NULL,*poly2=NULL;

    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d %d",&coeff,&exp);
        poly1=insert(poly1,coeff,exp);
    }
    scanf("%d",&m);
    for(int i=0;i<m;i++){
        scanf("%d %d",&coeff,&exp);
        poly2=insert(poly2,coeff,exp);
    }

    printf("Polynomial 1: ");
    displayPoly(poly1);

    printf("Polynomial 2: ");
    displayPoly(poly2);

    if(areEqual(poly1,poly2))
        printf("Polynomials are Equal.\n");
    else
        printf("Polynomials are not Equal.\n");
    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>

#define MAX_TERMS 100

struct Term{
   int coeff;
   int expo;
};

void addTerm(struct Term result[],int *size,int coeff,int expo){
   for(int i=0;i<*size;i++){
      if(result[i].expo==expo){
         result[i].coeff+=coeff;
         return;
      }
   }
   result[*size].coeff=coeff;
   result[*size].expo=expo;
   (*size)++;
}

void display(struct Term poly[],int size){
   for(int i=0;i<size;i++){
      printf("%d",poly[i].coeff);
      if(poly[i].expo==1) printf("x");
      else if(poly[i].expo!=0) printf("x^%d",poly[i].expo);
      if(i!=size-1) printf(" + ");
   }
   printf("\n");
}

void deleteTerm(struct Term poly[],int *size,int expo){
   for(int i=0;i<*size;i++){
      if(poly[i].expo==expo){
         for(int j=i;j<*size-1;j++){
            poly[j]=poly[j+1];
         }
```

```
            (*size)--;
            return;
        }
    }
}

int main(){
    int n,m,delexpo;
    struct Term poly1[10],poly2[10],result[MAX_TERMS];
    int resize=0;

    scanf("%d",&n);

    for(int i=0;i<n;i++)
        scanf("%d %d",&poly1[i].coeff,&poly1[i].expo);

    scanf("%d",&m);

    for(int i=0;i<m;i++)
        scanf("%d %d",&poly2[i].coeff,&poly2[i].expo);

    scanf("%d",&delexpo);

    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            int coeff=poly1[i].coeff * poly2[j].coeff;
            int expo=poly1[i].expo + poly2[j].expo;
            addTerm(result,&resize,coeff,expo);
        }
    }
    printf("Result of the multiplication: ");
    display(result, resize);

    deleteTerm(result, &resize, delexpo);

    printf("Result after deleting the term: ");
    display(result, resize);
    return 0;
}
```

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

3. Problem Statement

Rani is studying polynomials in her class. She has learned about polynomial multiplication and is eager to try it out on her own. However, she finds the process of manually multiplying polynomials quite tedious. To make her task easier, she decides to write a program to multiply two polynomials represented as linked lists.

Help Rani by designing a program that takes two polynomials as input and outputs their product polynomial. Each polynomial is represented by a linked list of terms, where each term has a coefficient and an exponent. The terms are entered in descending order of exponents.

***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 third line of output prints the resulting polynomial after multiplying the given polynomials.

The polynomials should be displayed in the format, where each term is represented as ax^b, where a is the coefficient and b is the exponent.

Refer to the sample output for the exact format.

***Sample Test Case***

Input: 2
2 3
3 2
2
3 2
2 1

Output: 2x^3 + 3x^2
3x^2 + 2x
6x^5 + 13x^4 + 6x^3

***Answer***

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

struct Term{
   int coeff,expo;
   struct Term* next;
};

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

void insertTerm(struct Term** poly,int c,int e){
   struct Term* newNode=createNode(c, e);
   if(*poly==NULL || (*poly)->expo < e){
      newNode->next=*poly;
      *poly=newNode;
      return;
   }

   struct Term* curr=*poly;
   struct Term* prev=NULL;

   while(curr && curr->expo > e){
```

```
        prev=curr;
        curr=curr->next;
    }
    if(curr && curr->expo==e){
        curr->coeff+=c;
        free(newNode);
    }else{
        newNode->next=curr;
        if(prev) prev->next=newNode;
    }
}

void multiply(struct Term* poly1,struct Term* poly2,struct Term** result){
    for(struct Term* p1=poly1;p1;p1=p1->next){
        for(struct Term* p2=poly2;p2;p2=p2->next){
            insertTerm(result, p1->coeff * p2->coeff,p1->expo + p2->expo);
        }
    }
}

void display(struct Term* poly){
    int first=0;
    while(poly){
        if(!first) printf(" + ");
        if(poly->expo==0) printf("%d",poly->coeff);
        else if(poly->expo==1) printf("%dx",poly->coeff);
        else printf("%dx^%d",poly->coeff,poly->expo);
        first =0;
        poly=poly->next;
    }
    printf("\n");
}

void readpoly(struct Term** poly,int n){
    int c,e;
    for(int i=0;i<n;i++){
        scanf("%d %d",&c ,&e);
        insertTerm(poly,c,e);
    }
}

int main(){
```

```
    int n,m;
    struct Term *poly1=NULL, *poly2=NULL, *result=NULL;

    scanf("%d",&n);
    readpoly(&poly1,n);

    scanf("%d",&m);
    readpoly(&poly2,m);

    multiply(poly1,poly2,&result);
    display(poly1);
    display(poly2);
    display(result);

    return 0;
}
```

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