Rajalakshmi Engineering College

Name: Nikhil Vinayak P
Email: 240701359@rajalakshmi.edu.in
Roll no: 240701359
Phone: 9884558531
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

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

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct Node {
```

```
    int coeff, 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 appendTerm(struct Node** poly, int coeff, int expo)
{
    if (coeff==0) return;
    struct Node* newNode=createNode(coeff, expo);
    if (*poly==NULL)
    {
        *poly=newNode;
    }
    else
    {
        struct Node* temp=*poly;
        while (temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newNode;
    }
}
void mergeLikeTerms(struct Node* poly)
{
    struct Node *ptr1=poly;
    while  (ptr1!=NULL)
    {
        struct Node *ptr2=ptr1;
        while (ptr2->next!=NULL)
        {
            if (ptr1->expo==ptr2->next->expo)
            {
                ptr1->coeff+=ptr2->next->coeff;
                struct Node* dup=ptr2->next;
                ptr2->next=ptr2->next->next;
```

```
            free(dup);
        }
        else
        {
            ptr2=ptr2->next;
        }
    }
    ptr1=ptr1->next;
  }
}
struct Node* multiplyPoly(struct Node* poly1, struct Node* poly2)
{
  struct Node* result=NULL;
  for (struct Node* i=poly1; i!=NULL; i=i->next)
  {
    for (struct Node* j=poly2; j!=NULL; j=j->next)
    {
      int coeff=i->coeff*j->coeff;
      int expo=i->expo+j->expo;
      appendTerm(&result, coeff, expo);
    }
  }
  mergeLikeTerms(result);
  return result;
}
void deleteTerm(struct Node** poly, int expo)
{
  struct Node *temp=*poly, *prev=NULL;
  while (temp!=NULL)
  {
    if (temp->expo==expo)
    {
      if (prev==NULL)
      {
        *poly=temp->next;
      }
      else
      {
        prev->next=temp->next;
      }
      free(temp);
      return;
```

```
        }
        prev=temp;
        temp=temp->next;
    }
}
void display(struct Node* poly)
{
    while (poly!=NULL)
    {
        if (poly->coeff==0)
        {
            poly=poly->next;
            continue;
        }
        if (poly!=NULL && poly!=poly->next)
        {
            if (poly!=NULL && poly!=poly->next)
            {
                if (poly!=NULL && poly->next!=NULL)
                {
                    printf("%d", poly->coeff);
                }
                else
                {
                    printf("%d", poly->coeff);
                }
            }
        }
        if (poly->expo>1)
        {
            printf("x^%d", poly->expo);
        }
        else if (poly->expo==1)
        {
            printf("x");
        }
        if (poly->next!=NULL)
        {
            printf(" + ");
        }
        poly=poly->next;
    }
```

```
    printf("\n");
}
int main()
{
    int n,m,coeff,expo,delExpo;
    struct Node *poly1=NULL, *poly2=NULL, *result=NULL;
    scanf("%d", &n);
    for (int i=0; i<n; i++)
    {
        scanf("%d %d", &coeff, &expo);
        appendTerm(&poly1, coeff, expo);
    }
    scanf("%d", &m);
    for (int i=0; i<m; i++)
    {
        scanf("%d %d", &coeff, &expo);
        appendTerm(&poly2, coeff, expo);
    }
    scanf("%d", &delExpo);
    result=multiplyPoly(poly1, poly2);
    printf("Result of the multiplication: ");
    display(result);
    deleteTerm(&result, delExpo);
    printf("Result after deleting the term: ");
    display(result);
    return 0;
}
```

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

2. 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 Term
{
    int coeff;
    int expo;
    struct Term* next;
} Term;
Term* appendTerm(Term* head, int coeff, int expo)
{
    Term* newNode=(Term*)malloc(sizeof(Term));
    newNode->coeff=coeff;
    newNode->expo=expo;
    newNode->next=NULL;
    if (head==NULL)
    {
        return newNode;
    }
    Term* temp=head;
    while (temp->next)
    {
        temp=temp->next;
    }
    temp->next=newNode;
    return head;
}
void displayPolynomial(Term* head)
{
    while(head)
    {
        printf("(%dx^%d)", head->coeff, head->expo);
        if (head->next)
        {
            printf(" + ");
        }
        head=head->next;
    }
    printf("\n");
}
int comparePolynomials(Term* p1, Term* p2)
```

```
{
    while (p1 && p2)
    {
        if (p1->expo!=p2->expo || p1->coeff!=p2->coeff)
        {
            return 0;
        }
        p1=p1->next;
        p2=p2->next;
    }
    return p1==NULL && p2==NULL;
}
void freePolynomial(Term* head)
{
    while (head)
    {
        Term* temp=head;
        head=head->next;
        free(temp);
    }
}
int main()
{
    int n,m,coeff,expo;
    Term *poly1=NULL, *poly2=NULL;
    scanf("%d", &n);
    for (int i=0; i<n; i++)
    {
        scanf("%d %d", &coeff, &expo);
        poly1=appendTerm(poly1, coeff, expo);
    }
    scanf("%d", &m);
    for (int i=0; i<m; i++)
    {
        scanf("%d %d", &coeff, &expo);
        poly2=appendTerm(poly2, coeff, expo);
    }
    printf("Polynomial 1: ");
    displayPolynomial(poly1);
    printf("Polynomial 2: ");
    displayPolynomial(poly2);
    if (comparePolynomials(poly1,poly2))
```

```
    {
        printf("Polynomials are Equal.\n");
    }
    else
    {
        printf("Polynomials are Not Equal.\n");
    }
    freePolynomial(poly1);
    freePolynomial(poly2);
    return 0;
}
```

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

3. Problem Statement

Timothy wants to evaluate polynomial expressions for his mathematics homework. He needs a program that allows him to input the coefficients of a polynomial based on its degree and compute the polynomial's value for a given input of x. Implement a function that takes the degree, coefficients, and the value of x, and returns the evaluated result of the polynomial.

Example

Input:

degree of the polynomial = 2

coefficient of x2 = 13

coefficient of x1 = 12

coefficient of x0 = 11

x = 1

Output:

36

Explanation:

Calculate the value of 13x2: 13 * 12 = 13.

Calculate the value of 12x1: 12 * 11 = 12.

Calculate the value of 11x0: 11 * 10 = 11.

Add the values of x2, x1, and x0 together: 13 + 12 + 11 = 36.

Input Format

The first line of input consists of an integer representing the degree of the polynomial.

The second line consists of an integer representing the coefficient of x2.

The third line consists of an integer representing the coefficient of x1.

The fourth line consists of an integer representing the coefficient of x0.

The fifth line consists of an integer representing the value of x, at which the polynomial should be evaluated.

Output Format

The output is an integer value obtained by evaluating the polynomial at the given value of x.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
13
12
11
1

Output: 36

Answer

```
// You are using GCC
#include <stdio.h>
#include <math.h>
int main()
{
```

```
    int degree;
    scanf("%d", &degree);
    int coefficients[degree+1];
    for (int i=degree; i>=0; i--)
    {
        scanf("%d", &coefficients[i]);
    }
    int x;
    scanf("%d", &x);
    int result=0;
    for (int i=0; i<=degree; i++)
    {
        result+=coefficients[i]*pow(x, i);
    }
    printf("%d\n", result);
    return 0;
}
```

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