

Rajalakshmi Engineering College

Name: Subhalakshmi M
Email: 240701539@rajalakshmi.edu.in
Roll no: 240701539
Phone: 6379032776
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 : 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>
```

```
struct Term{
    int coeff;
    int exp;
    struct Term*next;
};
typedef struct Term Term;
```

```
Term* createTerm(int coeff,int exp){
    Term* newTerm = (Term*)malloc(sizeof(Term));
    newTerm->coeff=coeff;
    newTerm->exp=exp;
    newTerm->next=NULL;
    return newTerm;
}
```

```
void insertTerm(Term ** poly,int coeff,int exp){
    Term* newTerm = createTerm(coeff,exp);
    if(*poly==NULL){
        *poly=newTerm;
    }else{
        Term*temp=*poly;
        while(temp->next)temp=temp->next;
        temp->next=newTerm;
    }
}
```

```
void displayPoly(Term* poly,const char* label){
    printf("%s: ",label);
    if(!poly){
        printf("0\n");
        return;
    }
```

```
    while(poly){
        printf("(%dx^%d)",poly->coeff,poly->exp);

        if(poly->next) printf(" + ");
        poly=poly->next;
    }
```

```

    printf("\n");
}

int compare(Term* poly1, Term* 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);
}

```

```

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

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

    displayPoly(poly1,"Polynomial 1");
    displayPoly(poly2,"Polynomial 2");

    if(compare(poly1,poly2)){
        printf("Polynomials are Equal.\n");
    }
    else{
        printf("Polynomials are Not Equal.\n");
    }
    return 0;
}

```

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 ' $cx^e + cx^e + \dots$ ' (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 coef;
    int expo;
    struct Node* next;
} Node;
```

```
void insert(Node** head, int coef, int expo) {
    if (coef == 0) return;
```

```
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->coef = coef;
    newNode->expo = expo;
    newNode->next = NULL;
```

```
    if (*head == NULL || (*head)->expo < expo) {
        newNode->next = *head;
        *head = newNode;
        return;
    }
```

```
    Node* temp = *head;
    Node* prev = NULL;
```

```
    while (temp != NULL && temp->expo > expo) {
        prev = temp;
        temp = temp->next;
    }
```

```
    if (temp != NULL && temp->expo == expo) {
        temp->coef += coef;
        free(newNode);
```

```
        if (temp->coef == 0) {
            if (prev == NULL) {
```

```

        *head = temp->next;
    } else {
        prev->next = temp->next;
    }
    free(temp);
}
return;
}

```

```

newNode->next = temp;
if (prev != NULL) {
    prev->next = newNode;
} else {
    *head = newNode;
}
}

```

```

void printpoly(Node* head) {
    if (head == NULL) {
        printf("0x^0\n");
        return;
    }
}

```

```

Node* temp = head;
int first = 1;

```

```

while (temp != NULL) {
    if (!first && temp->coef > 0) printf(" + ");
    if (temp->coef < 0) {
        if (!first) printf(" - ");
        else printf(" - ");
        printf("%dx^%d", -temp->coef, temp->expo);
    } else {
        printf("%dx^%d", temp->coef, temp->expo);
    }
}

```

```

    first = 0;
    temp = temp->next;
}
printf("\n");
}

```

```

int main() {
    int n, coef, expo;
    Node* originalHead = NULL, * simplifiedHead = NULL, * tail = NULL;

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d %d", &coef, &expo);

        // Build original polynomial
        Node* newNode = (Node*)malloc(sizeof(Node));
        newNode->coef = coef;
        newNode->expo = expo;
        newNode->next = NULL;

        if (originalHead == NULL) {
            originalHead = newNode;
            tail = newNode;
        } else {
            tail->next = newNode;
            tail = newNode;
        }

        // Build simplified polynomial
        insert(&simplifiedHead, coef, expo);
    }

    printf("Original polynomial: ");
    printpoly(originalHead);
    printf("Simplified polynomial: ");
    printpoly(simplifiedHead);

    return 0;
}

```

Status : Wrong

Marks : 0/10

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 x^2 = 13

coefficient of x^1 = 12

coefficient of x^0 = 11

x = 1

Output:

36

Explanation:

Calculate the value of $13x^2$: $13 * 12 = 13$.

Calculate the value of $12x^1$: $12 * 11 = 12$.

Calculate the value of $11x^0$: $11 * 10 = 11$.

Add the values of x^2 , x^1 , and x^0 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 x^2 .

The third line consists of an integer representing the coefficient of x^1 .

The fourth line consists of an integer representing the coefficient of x^0 .

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<stdlib.h>
```

```
#include<math.h>
```

```
struct Node{  
    int coeff;  
    int power;  
    struct Node* next;  
};
```

```
struct Node* head=NULL;
```

```
void insertTerm(int coeff,int power){  
    struct Node*newNode = (struct Node*)malloc(sizeof(struct Node));  
    newNode->coeff=coeff;  
    newNode->power=power;  
    newNode->next=NULL;
```

```
    if(!head){  
        head=newNode;
```

```
    }
```

```
    else{  
        struct Node*temp=head;  
        while(temp->next){
```

```

        temp=temp->next;
    }
    temp->next=newNode;
}
}
int evaluvatePolynomial(int x){
    int result=0;
    struct Node* temp=head;

    while(temp){
        result+=temp->coeff*pow(x,temp->power);
        temp=temp->next;
    }
    return result;
}
void displayPolynomial(){
    struct Node* temp=head;

    while(temp){
        printf("%dx^%d ",temp->coeff,temp->power);
        if(temp->next) printf(" + ");
        temp=temp->next;
    }
    printf("\n");
}
int main(){
    int degree,x,coeff;
    scanf("%d",&degree);
    for(int i=degree;i>=0;i--){
        scanf("%d",&coeff);
        insertTerm(coeff,i);
    }

    scanf("%d",&x);
    int result=evaluvatePolynomial(x);
    printf("%d",result);

    return 0;
}

```

Status : Correct

Marks : 10/10