Rajalakshmi Engineering College

Name: srivatsen s
Email: 240701534@rajalakshmi.edu.in
Roll no: 2116240701534
Phone: 9042122714
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 : 30

Section 1 : Coding

1. 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 coeff2, coeff1, coeff0;
    scanf("%d", &coeff2);
    scanf("%d", &coeff1);
    scanf("%d", &coeff0);
    int x;
    scanf("%d", &x);
    int result = coeff2 * pow(x, 2) + coeff1 * x + coeff0;
    printf("%d\n", result);
    return 0;
}
```

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

2. 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

```
// 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* insertTerm(Node* head, int coeff, int exp) {
    if (coeff == 0) return head;

    Node* newNode = createNode(coeff, exp);
    if (!head || exp < head->exp) {
        newNode->next = head;
        return newNode;
    }

    Node* current = head;
    Node* prev = NULL;
    while (current && current->exp < exp) {
        prev = current;
        current = current->next;
    }

    if (current && current->exp == exp) {
        current->coeff += coeff;
        free(newNode);
        if (current->coeff == 0) {
            if (prev) prev->next = current->next;
            else head = current->next;
            free(current);
        }
        return head;
    }

    newNode->next = current;
    if (prev) prev->next = newNode;
    return head;
```

```
}

// Read a polynomial from input
Node* readPolynomial() {
    Node* head = NULL;
    int coeff, exp;
    while (1) {
        scanf("%d %d", &coeff, &exp);
        if (coeff == 0 && exp == 0) break;
        head = insertTerm(head, coeff, exp);
    }
    return head;
}

Node* addPolynomials(Node* poly1, Node* poly2) {
    Node* result = NULL;
    Node* p1 = poly1;
    Node* p2 = poly2;

    while (p1) {
        result = insertTerm(result, p1->coeff, p1->exp);
        p1 = p1->next;
    }

    while (p2) {
        result = insertTerm(result, p2->coeff, p2->exp);
        p2 = p2->next;
    }

    return result;
}

void printPolynomial(Node* head) {
    if (!head) {
        printf("0\n");
        return;
    }

    Node* temp = head;
    while (temp) {
        printf("%dx^%d", temp->coeff, temp->exp);
        if (temp->next) printf(" + ");
```

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

void freePolynomial(Node* head) {
    while (head) {
        Node* temp = head;
        head = head->next;
        free(temp);
    }
}

int main() {
    Node *poly1, *poly2, *sum;

    poly1 = readPolynomial();
    poly2 = readPolynomial();
    sum = addPolynomials(poly1, poly2);

    printPolynomial(poly1);
    printPolynomial(poly2);
    printPolynomial(sum);

    freePolynomial(poly1);
    freePolynomial(poly2);
    freePolynomial(sum);

    return 0;
}
```

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>

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

// Function to display polynomial
void displayPolynomial(Node* head) {
    Node* current = head;
    while (current != NULL) {
        printf("(%dx^%d)", current->coeff, current->exp);
        if (current->next != NULL)
            printf(" + ");
```

```
        current = current->next;
    }
    printf("\n");
}

// Function to compare two polynomials
int areEqual(Node* p1, Node* p2) {
    while (p1 != NULL && p2 != NULL) {
        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 = insertTerm(poly1, coeff, exp);
    }

    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &coeff, &exp);
        poly2 = insertTerm(poly2, coeff, exp);
    }
    printf("Polynomial 1: ");
    displayPolynomial(poly1);

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

    if (areEqual(poly1, poly2))
        printf("Polynomials are Equal.\n");
    else
        printf("Polynomials are Not Equal.\n");
    return 0;
```

}

Status : Correct *Marks : 10/10*