Rajalakshmi Engineering College

Name: V Naresh Kumar
Email: 241501260@rajalakshmi.edu.in
Roll no: 241501260
Phone: 9080390434
Branch: REC
Department: I AI & ML FC
Batch: 2028
Degree: B.E - AI & ML

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

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

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

#define MAX_EXP 101

// Structure for a term in the polynomial
struct Term {
    int coeff;
    int exp;
    struct Term* next;
};

// Function to create a new term
struct Term* createTerm(int coeff, int exp) {
    struct Term* newTerm = (struct Term*)malloc(sizeof(struct Term));
```

```
    newTerm->coeff = coeff;
    newTerm->exp = exp;
    newTerm->next = NULL;
    return newTerm;
}

// Append term to the list
void appendTerm(struct Term** head, int coeff, int exp) {
    struct Term* newNode = createTerm(coeff, exp);
    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Term* temp = *head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }
}

// Print the polynomial
void printPolynomial(struct Term* head) {
    struct Term* temp = head;
    int first = 1;
    while (temp != NULL) {
        if (!first) printf(" + ");
        printf("%dx^%d", temp->coeff, temp->exp);
        temp = temp->next;
        first = 0;
    }
    if (head == NULL) {
        printf("0");
    }
    printf("\n");
}

// Simplify polynomial
struct Term* simplify(struct Term* head) {
    int coeffs[MAX_EXP] = {0};

    // Sum coefficients with the same exponent
    struct Term* temp = head;
    while (temp != NULL) {
```

```
        coeffs[temp->exp] += temp->coeff;
        temp = temp->next;
    }

    // Create simplified list in descending order of exponent
    struct Term* simplified = NULL;
    for (int i = MAX_EXP - 1; i >= 0; i--) {
        if (coeffs[i] != 0) {
            appendTerm(&simplified, coeffs[i], i);
        }
    }

    // Special case: all zero terms
    if (simplified == NULL) {
        appendTerm(&simplified, 0, 0);
    }

    return simplified;
}

int main() {
    int n;
    scanf("%d", &n);

    struct Term* original = NULL;
    for (int i = 0; i < n; i++) {
        int coeff, exp;
        scanf("%d %d", &coeff, &exp);
        appendTerm(&original, coeff, exp);
    }

    printf("Original polynomial: ");
    printPolynomial(original);

    struct Term* simplified = simplify(original);
    printf("Simplified polynomial: ");
    printPolynomial(simplified);

    return 0;
}
```

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

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>

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

// Function to create a new term node
struct Term* createTerm(int coeff, int exp) {
    struct Term* newTerm = (struct Term*)malloc(sizeof(struct Term));
    newTerm->coeff = coeff;
    newTerm->exp = exp;
    newTerm->next = NULL;
    return newTerm;
}

// Function to insert a term in sorted order (ascending by exponent)
void insertTerm(struct Term** poly, int coeff, int exp) {
    if (coeff == 0) return;  // Skip zero coefficient

    struct Term* newTerm = createTerm(coeff, exp);
    if (*poly == NULL || (*poly)->exp > exp) {
        newTerm->next = *poly;
        *poly = newTerm;
        return;
    }

    struct Term* current = *poly;
```

```
    struct Term* prev = NULL;
    while (current != NULL && current->exp < exp) {
        prev = current;
        current = current->next;
    }

    // Combine if same exponent
    if (current != NULL && current->exp == exp) {
        current->coeff += coeff;
        free(newTerm);
        if (current->coeff == 0) {
            // Remove the term if coefficient becomes 0
            if (prev == NULL) {
                *poly = current->next;
            } else {
                prev->next = current->next;
            }
            free(current);
        }
    } else {
        newTerm->next = current;
        prev->next = newTerm;
    }
}

// Read polynomial until 0 0
struct Term* readPolynomial() {
    int coeff, exp;
    struct Term* poly = NULL;
    while (1) {
        scanf("%d %d", &coeff, &exp);
        if (coeff == 0 && exp == 0) break;
        insertTerm(&poly, coeff, exp);
    }
    return poly;
}

// Add two polynomials
struct Term* addPolynomials(struct Term* poly1, struct Term* poly2) {
    struct Term* result = NULL;

    // Copy poly1
```

```
    struct Term* temp1 = poly1;
    while (temp1 != NULL) {
        insertTerm(&result, temp1->coeff, temp1->exp);
        temp1 = temp1->next;
    }

    // Add poly2
    struct Term* temp2 = poly2;
    while (temp2 != NULL) {
        insertTerm(&result, temp2->coeff, temp2->exp);
        temp2 = temp2->next;
    }

    return result;
}

// Print polynomial
void printPolynomial(struct Term* poly) {
    if (poly == NULL) {
        printf("0\n");
        return;
    }

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

// Free memory
void freePolynomial(struct Term* poly) {
    while (poly != NULL) {
        struct Term* temp = poly;
        poly = poly->next;
        free(temp);
    }
}
```

```
int main() {
    struct Term* poly1 = readPolynomial();
    struct Term* poly2 = readPolynomial();
    struct Term* result = addPolynomials(poly1, poly2);

    printPolynomial(poly1);
    printPolynomial(poly2);
    printPolynomial(result);

    freePolynomial(poly1);
    freePolynomial(poly2);
    freePolynomial(result);

    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

```c
// You are using GCC
#include <stdio.h>
#include <math.h>

int main() {
    int degree;
    int a, b, c; // coefficients for x^2, x^1, x^0
    int x, result;

    // Read the degree of the polynomial (should always be 2 as per constraints)
    scanf("%d", &degree);

    // Read coefficients for x^2, x^1, and x^0
    scanf("%d", &a);  // Coefficient of x^2
    scanf("%d", &b);  // Coefficient of x^1
    scanf("%d", &c);  // Coefficient of x^0

    // Read the value of x
    scanf("%d", &x);

    // Calculate the polynomial value using the formula: a*x^2 + b*x + c
    result = a * pow(x, 2) + b * x + c;

    // Print the result
    printf("%d\n", result);

    return 0;
}
```

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