Rajalakshmi Engineering College

Name: NISHANTH ELANGO RAJAN
Email: 241901075@rajalakshmi.edu.in
Roll no: 241901075
Phone: 9444909050
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

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

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

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

struct Term {
    int coefficient;
    int exponent;
```

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

void insertTerm(struct Term **poly, int coefficient, int exponent) {
    struct Term *newTerm = createTerm(coefficient, exponent);

    if (*poly == NULL || (*poly)->exponent > exponent) {
        newTerm->next = *poly;
        *poly = newTerm;
    } else {
        struct Term *temp = *poly;
        while (temp->next != NULL && temp->next->exponent < exponent) {
            temp = temp->next;
        }
        if (temp->next != NULL && temp->next->exponent == exponent) {
            temp->next->coefficient += coefficient;
            free(newTerm);
        } else {
            newTerm->next = temp->next;
            temp->next = newTerm;
        }
    }
}
struct Term* readPolynomial() {
    struct Term *poly = NULL;
    int coefficient, exponent;

    while (1) {
        scanf("%d %d", &coefficient, &exponent);
        if (coefficient == 0 && exponent == 0) {
            break;
        }
        insertTerm(&poly, coefficient, exponent);
    }
    return poly;
```

```
}
void printPolynomial(struct Term *poly) {
    if (poly == NULL) {
        printf("0\n");
        return;
    }
    int first = 1;
    while (poly != NULL) {
        if (first) {
            printf("%dx^%d", poly->coefficient, poly->exponent);
            first = 0;
        } else {
            printf(" + %dx^%d", poly->coefficient, poly->exponent);
        }
        poly = poly->next;
    }
    printf("\n");
}

struct Term* addPolynomials(struct Term *poly1, struct Term *poly2) {
    struct Term *result = NULL;
    while (poly1 != NULL && poly2 != NULL) {
        if (poly1->exponent < poly2->exponent) {
            insertTerm(&result, poly1->coefficient, poly1->exponent);
            poly1 = poly1->next;
        } else if (poly1->exponent > poly2->exponent) {
            insertTerm(&result, poly2->coefficient, poly2->exponent);
            poly2 = poly2->next;
        } else {
            int sum = poly1->coefficient + poly2->coefficient;
            if (sum != 0) {
                insertTerm(&result, sum, poly1->exponent);
            }
            poly1 = poly1->next;
            poly2 = poly2->next;
        }
    }
    while (poly1 != NULL) {
        insertTerm(&result, poly1->coefficient, poly1->exponent);
        poly1 = poly1->next;
    }
    while (poly2 != NULL) {
```

```
        insertTerm(&result, poly2->coefficient, poly2->exponent);
        poly2 = poly2->next;
    }
    return result;
}

int main() {
    struct Term *poly1 = readPolynomial();
    struct Term *poly2 = readPolynomial();
    printPolynomial(poly1);
    printPolynomial(poly2);
    struct Term *result = addPolynomials(poly1, poly2);
    printPolynomial(result);
    return 0;
}
```

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

2. Problem Statement

Hasini is studying polynomials in her class. Her teacher has introduced a new concept of two polynomials using linked lists.

The teacher provides Hasini with a program that takes two polynomials as input, represented as linked lists, and then displays them together. The polynomials are simplified and should be displayed in the format ax^b, where a is the coefficient and b is the exponent.

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 polynomials should be displayed in the format ax^b, where a is the coefficient and b is the exponent.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 2
2 1
3 0
3
2 2
1 1
4 0
Output: 1x^2 + 2x + 3
2x^2 + 1x + 4

Answer

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

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

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

```
}

void insertTerm(struct Term** poly, int coef, int exp) {
    struct Term* newTerm = createTerm(coef, exp);
    if (*poly == NULL || (*poly)->exp < exp) {
        newTerm->next = *poly;
        *poly = newTerm;
    } else {
        struct Term* temp = *poly;
        while (temp->next != NULL && temp->next->exp > exp) {
            temp = temp->next;
        }
        if (temp->next != NULL && temp->next->exp == exp) {
            temp->next->coef += coef;
            free(newTerm);
        } else {
            newTerm->next = temp->next;
            temp->next = newTerm;
        }
    }
}

void printPolynomial(struct Term* poly) {
    if (poly == NULL) {
        printf("0");
        return;
    }

    int firstTerm = 1;
    while (poly != NULL) {
        if (poly->coef != 0) {
            if (!firstTerm) {
                if (poly->coef > 0) {
                    printf(" + ");
                } else {
                    printf(" - ");
                }
            }

            if (firstTerm) {
                firstTerm = 0;
                if (poly->coef < 0) {
```

```
            printf("-");
        }
    }

    if (poly->exp == 0) {
        printf("%d", abs(poly->coef));
    } else if (poly->exp == 1) {
        printf("%dx", abs(poly->coef));
    } else {
        printf("%dx^%d", abs(poly->coef), poly->exp);
    }
  }
  poly = poly->next;
 }
 printf("\n");
}

int main() {
  int n, m, coef, exp;

  struct Term* poly1 = NULL;
  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
    scanf("%d %d", &coef, &exp);
    insertTerm(&poly1, coef, exp);
  }

  struct Term* poly2 = NULL;
  scanf("%d", &m);
  for (int i = 0; i < m; i++) {
    scanf("%d %d", &coef, &exp);
    insertTerm(&poly2, coef, exp);
  }
  printPolynomial(poly1);
  printPolynomial(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

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

int evaluatePolynomial(int degree, int coeffs[], int x) {
    int result = 0;

    for (int i = 0; i <= degree; i++) {
        result += coeffs[i] * (int)pow(x, degree - i); // Use the formula to evaluate the
polynomial
    }

    return result;
}

int main() {
    int degree;

    scanf("%d", &degree);

    int coeffs[degree + 1];
```

```
    for (int i = 0; i <= degree; i++) {
        scanf("%d", &coeffs[i]);
    }

    int x;
    scanf("%d", &x);

    int result = evaluatePolynomial(degree, coeffs, x);

    printf("%d\n", result);

    return 0;
}
```

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