Rajalakshmi Engineering College

Name: Priyanka S
Email: 241801217@rajalakshmi.edu.in
Roll no: 2116241801217
Phone: 8072623531
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 28.5

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

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

typedef struct {
    int coeff;
    int exp;
} Term;

Term* multiplyPolynomials(Term* poly1, int size1, Term* poly2, int size2, int*
resultSize) {
    int maxSize = size1 * size2;
    Term* result = (Term*)malloc(maxSize * sizeof(Term));
    *resultSize = 0;

    for (int i = 0; i < size1; i++) {
        for (int j = 0; j < size2; j++) {
            int newCoeff = poly1[i].coeff * poly2[j].coeff;
            int newExp = poly1[i].exp + poly2[j].exp;

            int found = 0;
            for (int k = 0; k < *resultSize; k++) {
                if (result[k].exp == newExp) {
                    result[k].coeff += newCoeff;
                    found = 1;
                    break;
                }
            }

            if (!found) {
                result[*resultSize].coeff = newCoeff;
                result[*resultSize].exp = newExp;
                (*resultSize)++;
            }
        }
    }

    Term* finalResult = (Term*)malloc(*resultSize * sizeof(Term));
    for (int i = 0; i < *resultSize; i++) {
        finalResult[i] = result[i];
    }
    free(result);
    return finalResult;
}

Term* deleteTerm(Term* poly, int* size, int expToDelete) {
    int newSize = 0;
    Term* newPoly = (Term*)malloc(*size * sizeof(Term));

    for (int i = 0; i < *size; i++) {
        if (poly[i].exp != expToDelete) {
            newPoly[newSize++] = poly[i];
        }
    }

    free(poly);
    *size = newSize;
    return (Term*)realloc(newPoly, newSize * sizeof(Term));
}

void printPolynomial(Term* poly, int size) {
    if (size == 0) {
        printf("0\n");
        return;
    }

    for (int i = 0; i < size; i++) {
        if (poly[i].coeff != 0) {
            if (i > 0 && poly[i].coeff > 0) {
                printf(" + ");
            } else if (i > 0 && poly[i].coeff < 0){
                printf(" - ");
                poly[i].coeff = abs(poly[i].coeff);
            }
            if (poly[i].coeff != 1 || poly[i].exp == 0){
                printf("%d", poly[i].coeff);
            }
            if (poly[i].exp != 0) {
                printf("x");
                if (poly[i].exp != 1) {
                    printf("^%d", poly[i].exp);
                }
            }
        }
    }
    printf("\n");
}

int main() {
    int size1, size2, resultSize, expToDelete;


    scanf("%d", &size1);
    Term* poly1 = (Term*)malloc(size1 * sizeof(Term));

    for (int i = 0; i < size1; i++) {
        scanf("%d %d", &poly1[i].coeff, &poly1[i].exp);
    }


    scanf("%d", &size2);
    Term* poly2 = (Term*)malloc(size2 * sizeof(Term));

    for (int i = 0; i < size2; i++) {
        scanf("%d %d", &poly2[i].coeff, &poly2[i].exp);
    }

    Term* resultPoly = multiplyPolynomials(poly1, size1, poly2, size2, &resultSize);

    printf("Result of the multiplication: ");
    printPolynomial(resultPoly, resultSize);

    printf("Result after deleting the term: ");
    scanf("%d", &expToDelete);

    resultPoly = deleteTerm(resultPoly, &resultSize, expToDelete);


    printPolynomial(resultPoly, resultSize);

    free(poly1);
    free(poly2);
    free(resultPoly);

    return 0;
}
```

Status : Partially correct ***Marks : 8.5/10***

2. 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 coeff[degree+1];
   for(int i=degree;i>=0;i--)
   {
      scanf("%d",&coeff[i]);
   }
   int x;
   scanf("%d",&x);
   int result=0;

   for(int i=degree;i>=0;i--)
   {
      result += coeff[i]*pow(x,i);
   }
   printf("%d",result);
```

```
    return 0;
}
```

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

3. 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>

typedef struct Term {
   int coefficient;
   int exponent;
   struct Term *next;
} Term;

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

void insertTerm(Term **head, int coefficient, int exponent) {
   Term *newTerm = createTerm(coefficient, exponent);
   if (*head == NULL || (*head)->exponent > exponent) {
      newTerm->next = *head;
      *head = newTerm;
   } else {
      Term *current = *head;
      while (current->next != NULL && current->next->exponent < exponent) {
            current = current->next;
        }
        if (current->next != NULL && current->next->exponent == exponent) {
            current->next->coefficient += coefficient;
            free(newTerm);
        } else {
            newTerm->next = current->next;
            current->next = newTerm;
        }
    }
}

void addPolynomials(Term *p1, Term *p2, Term **result) {
    while (p1 != NULL || p2 != NULL) {
        if (p1 == NULL) {
            insertTerm(result, p2->coefficient, p2->exponent);
            p2 = p2->next;
        } else if (p2 == NULL) {
            insertTerm(result, p1->coefficient, p1->exponent);
            p1 = p1->next;
        } else if (p1->exponent < p2->exponent) {
            insertTerm(result, p1->coefficient, p1->exponent);
            p1 = p1->next;
        } else if (p1->exponent > p2->exponent) {
            insertTerm(result, p2->coefficient, p2->exponent);
            p2 = p2->next;
        } else {
            insertTerm(result, p1->coefficient + p2->coefficient, p1->exponent);
            p1 = p1->next;
            p2 = p2->next;
        }
    }
}

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

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

int main() {
    Term *poly1 = NULL, *poly2 = NULL, *result = NULL;
    int coefficient, exponent;

    while (1) {
        scanf("%d %d", &coefficient, &exponent);
        if (coefficient == 0 && exponent == 0) break;
        insertTerm(&poly1, coefficient, exponent);
    }

    while (1) {
        scanf("%d %d", &coefficient, &exponent);
        if (coefficient == 0 && exponent == 0) break;
        insertTerm(&poly2, coefficient, exponent);
    }

    printPolynomial(poly1);
    printPolynomial(poly2);

    addPolynomials(poly1, poly2, &result);
    printPolynomial(result);

    freePolynomial(poly1);
    freePolynomial(poly2);
    freePolynomial(result);
    return 0;
}
```

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