Rajalakshmi Engineering College

Name: mahati sathish
Email: 240701302@rajalakshmi.edu.in
Roll no: 240701302
Phone: 9176485888
Branch: REC
Department: I CSE FC
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

Rani is studying polynomials in her class. She has learned about polynomial multiplication and is eager to try it out on her own. However, she finds the process of manually multiplying polynomials quite tedious. To make her task easier, she decides to write a program to multiply two polynomials represented as linked lists.

Help Rani by designing a program that takes two polynomials as input and outputs their product polynomial. Each polynomial is represented by a linked list of terms, where each term has a coefficient and an exponent. The terms are entered in descending order of exponents.

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 third line of output prints the resulting polynomial after multiplying the given polynomials.

The polynomials should be displayed in the format, where each term is represented as ax^b, where a is the coefficient and b is the exponent.

Refer to the sample output for the exact format.

Sample Test Case

Input: 2
2 3
3 2
2
3 2
2 1

Output: 2x^3 + 3x^2
3x^2 + 2x
6x^5 + 13x^4 + 6x^3

Answer

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

```
typedef struct Term
{

    int coeff;
    int exp;
    struct Term* next;

} Term;

Term* createTerm(int coeff, int exp)
{

    Term* newTerm = (Term*)malloc(sizeof(Term));
    newTerm->coeff = coeff;
    newTerm->exp = exp;
    newTerm->next = NULL;
    return newTerm;

}

Term* insertTerm(Term* head, int coeff, int exp)
{

    if (coeff == 0) return head;
    Term* newTerm = createTerm(coeff, exp);
    if (!head || exp > head->exp)
{

        newTerm->next = head;
        return newTerm;

}
```

```
    Term* temp = head;
    Term* prev = NULL;
    while (temp && exp < temp->exp)

{


        prev = temp;
        temp = temp->next;


}
    if (temp && temp->exp == exp)

{


        temp->coeff += coeff;
        if (temp->coeff == 0)

{


            if (prev) prev->next = temp->next;
            else head = temp->next;
            free(temp);


}
        free(newTerm);


} else

{


        newTerm->next = temp;
        if (prev) prev->next = newTerm;
        else head = newTerm;
```

```
}
    return head;
}

Term* multiplyPolynomials(Term* poly1, Term* poly2)

{


    Term* result = NULL;
    for (Term* t1 = poly1; t1; t1 = t1->next)

{


        for (Term* t2 = poly2; t2; t2 = t2->next)

{


            result = insertTerm(result, t1->coeff * t2->coeff, t1->exp + t2->exp);


}


}
    return result;

}

void printPolynomial(Term* head)

{


    Term* temp = head;
    while (temp)

{
```

```
        if (temp->coeff > 0 && temp != head)
            printf(" + ");
        if (temp->exp == 0)
            printf("%d", temp->coeff);
        else if (temp->exp == 1)
            printf("%dx", temp->coeff);
        else
            printf("%dx^%d", temp->coeff, temp->exp);
        temp = temp->next;


    }
        printf("\n");
    }

    Term* readPolynomial(int n)

    {


        Term* poly = NULL;
        for (int i = 0; i < n; i++)

    {


            int coeff, exp;
            scanf("%d %d", &coeff, &exp);
            poly = insertTerm(poly, coeff, exp);


    }
        return poly;

    }

    int main()

    {
```

```
    int n, m;
    scanf("%d", &n);
    Term* poly1 = readPolynomial(n);
    scanf("%d", &m);
    Term* poly2 = readPolynomial(m);
    Term* result = multiplyPolynomials(poly1, poly2);
    printPolynomial(poly1);
    printPolynomial(poly2);
    printPolynomial(result);
    return 0;

}
```

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

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>
```

```
int main()
{

    int degree, coeff2, coeff1, coeff0, x, result;

    scanf("%d", &degree);
    scanf("%d", &coeff2);
    scanf("%d", &coeff1);
    scanf("%d", &coeff0);
    scanf("%d", &x);

    result = coeff2 * x * x + coeff1 * x + coeff0;

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

    return 0;

}
```

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

3. Problem Statement

Lisa is studying polynomials in her class. She is learning about the multiplication of polynomials.

To practice her understanding, she wants to write a program that multiplies two polynomials and displays the result. Each polynomial is represented as a linked list, where each node contains the coefficient and exponent of a term.

Example

Input:

4 3

y

3 1

y

1 0

n

2 2

y

3 1

y

2 0

n

Output:

8x^5 + 12x^4 + 14x^3 + 11x^2 + 9x + 2

Explanation

1. Poly1: 4x^3 + 3x + 1
2. Poly2: 2x^2 + 3x + 2

Multiplication Steps:

1. Multiply 4x^3 by Poly2:
   -> 4x^3 * 2x^2 = 8x^5
   -> 4x^3 * 3x = 12x^4
   -> 4x^3 * 2 = 8x^3
2. Multiply 3x by Poly2:
   -> 3x * 2x^2 = 6x^3
   -> 3x * 3x = 9x^2
   -> 3x * 2 = 6x

3. Multiply 1 by Poly2:

-> 1 * 2x^2 = 2x^2

-> 1 * 3x = 3x

-> 1 * 2 = 2

Combine the results: 8x^5 + 12x^4 + (8x^3 + 6x^3) + (9x^2 + 2x^2) + (6x + 3x) + 2

The combined polynomial is: 8x^5 + 12x^4 + 14x^3 + 11x^2 + 9x + 2

Input Format

The input consists of two sets of polynomial terms.

Each polynomial term is represented by two integers separated by a space:

- The first integer represents the coefficient of the term.
- The second integer represents the exponent of the term.

After entering a polynomial term, the user is prompted to input a character indicating whether to continue adding more terms to the polynomial.

If the user inputs 'y' or 'Y', the program continues to accept more terms.

If the user inputs 'n' or 'N', the program moves on to the next polynomial.

Output Format

The output consists of a single line representing the resulting polynomial after multiplying the two input polynomials.

Each term of the resulting polynomial is formatted as follows:

- The coefficient and exponent are separated by 'x^' if the exponent is greater than 1.

- If the exponent is 1, only 'x' is displayed without the exponent.
- If the exponent is 0, only the coefficient is displayed.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4 3
y
3 1
y
1 0
n
2 2
y
3 1
y
2 0
n
Output: 8x^5 + 12x^4 + 14x^3 + 11x^2 + 9x + 2

Answer

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

typedef struct Term

{


    int coeff;
    int exp;
    struct Term* next;

} 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 appendTerm(Term** poly, int coeff, int exp)

{

    Term* newTerm = createTerm(coeff, exp);
    if (*poly == NULL)

{


        *poly = newTerm;


} else

{


        Term* temp = *poly;
        while (temp->next != NULL)

{


            temp = temp->next;


}
    temp->next = newTerm;
```

```
}
}
void multiplyPolynomials(Term* poly1, Term* poly2, Term** result)
{

   for (Term* p1 = poly1; p1 != NULL; p1 = p1->next)
{

      for (Term* p2 = poly2; p2 != NULL; p2 = p2->next)
{

         int coeff = p1->coeff * p2->coeff;
         int exp = p1->exp + p2->exp;
         appendTerm(result, coeff, exp);


}

}
}
void combineLikeTerms(Term** poly)
{

   for (Term* p1 = *poly; p1 != NULL; p1 = p1->next)
{

      for (Term* p2 = p1->next; p2 != NULL; p2 = p2->next)
```

```
{

        if (p1->exp == p2->exp)

{


            p1->coeff += p2->coeff;
            p2->coeff = 0;


}

}


}

}

void printPolynomial(Term* poly)

{

   int firstTerm = 1;
   for (Term* temp = poly; temp != NULL; temp = temp->next)

{


      if (temp->coeff == 0) continue;
      if (!firstTerm)

{


        if (temp->coeff > 0)
```

```
{
            printf(" + ");

} else
{

            printf(" - ");
            temp->coeff = -temp->coeff;

}

}
      if (temp->exp > 1)
{

          printf("%dx^%d", temp->coeff, temp->exp);

} else if (temp->exp == 1)
{

          printf("%dx", temp->coeff);

} else
{

          printf("%d", temp->coeff);
```

```
    }
        firstTerm = 0;


    }
       printf("\n");

    }

    int main()

    {


       Term* poly1 = NULL;
       Term* poly2 = NULL;
       Term* result = NULL;

       int coeff, exp;
       char cont;


       while (1)

    {


           scanf("%d %d", &coeff, &exp);
           appendTerm(&poly1, coeff, exp);

           scanf(" %c", &cont);
           if (cont == 'n' || cont == 'N') break;


    }


       while (1)

    {
```

```
        scanf("%d %d", &coeff, &exp);
        appendTerm(&poly2, coeff, exp);
        scanf(" %c", &cont);
        if (cont == 'n' || cont == 'N') break;


    }

    multiplyPolynomials(poly1, poly2, &result);
    combineLikeTerms(&result);
    printPolynomial(result);

    return 0;
}
```

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