

# Rajalakshmi Engineering College

Name: Siva.S Siva.S  
Email: 240701515@rajalakshmi.edu.in  
Roll no: 2116240701515  
Phone: 9940699577  
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 : 28

### 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  $x^2$  = 13

coefficient of  $x^1$  = 12

coefficient of  $x_0 = 11$

$x = 1$

Output:

36

Explanation:

Calculate the value of  $13x^2$ :  $13 * 12 = 13$ .

Calculate the value of  $12x_1$ :  $12 * 11 = 12$ .

Calculate the value of  $11x_0$ :  $11 * 10 = 11$ .

Add the values of  $x_2$ ,  $x_1$ , and  $x_0$  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  $x_2$ .

The third line consists of an integer representing the coefficient of  $x_1$ .

The fourth line consists of an integer representing the coefficient of  $x_0$ .

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;
    int coeff2, coeff1, coeff0;
    int x;
    int 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

## 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***

```
// You are using GCC
```

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

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

```
struct Node {  
    int coeff;  
    int exp;  
    struct Node*next;  
};
```

```
struct Node* createNode(int coeff, int exp) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    newNode->coeff = coeff;  
    newNode->exp = exp;  
    newNode->next = NULL;  
    return newNode;  
}
```

```
void insertNode(struct Node** head, int coeff, int exp) {  
    struct Node* newNode = createNode(coeff, exp);  
    if(*head == NULL) {  
        *head = newNode;  
    } else {  
        struct Node* temp = *head;  
        while (temp->next != NULL)  
            temp = temp->next;  
        temp->next = newNode;  
    }  
}
```

```
void displayPolynomial(struct Node* head) {  
    struct Node* temp = head;  
    int firstTerm = 1;  
    while (temp != NULL){  
        if(!firstTerm && temp->coeff >= 0)  
            printf(" + ");  
        else if (!firstTerm && temp->coeff < 0)  
            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);

        firstTerm = 0;
        temp = temp->next;
    }
    printf("\n");
}

```

```

int main() {
    int n, m, coeff, exp;
    struct Node* poly1 = NULL;
    struct Node* poly2 = NULL;

    scanf("%d ", &n);
    for(int i = 0; i < n; i++) {
        scanf("%d %d", &coeff, &exp);
        insertNode(&poly1, coeff, exp);
    }
    scanf("%d ", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &coeff, &exp);
        insertNode(&poly2, coeff, exp);
    }

    displayPolynomial(poly1);
    displayPolynomial(poly2);

    return 0;
}

```

**Status :** Partially correct

**Marks :** 8/10

### 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>
```

```
struct Node {
    int coeff;
    int exp;
    struct Node* next;
};
```

```
struct Node* createNode(int coeff, int exp) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->coeff = coeff;
    newNode->exp = exp;
    newNode->next = NULL;
    return newNode;
}
```

```
void insertSorted(struct Node** head, int coeff, int exp) {
    if (coeff == 0) return;
```

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

```
    struct Node* curr = *head;
    struct Node* prev = NULL;
    while (curr != NULL && curr->exp < exp) {
        prev = curr;
        curr = curr->next;
    }
```

```
    if (curr != NULL && curr->exp == exp) {
        curr->coeff += coeff;
        if (curr->coeff == 0) {
            if (prev == NULL) {
```



```

        *head = curr-> next;
    } else {
        prev->next = curr-> next;
    }
    free(curr);
}
free(newNode);
} else {
    newNode->next = curr;
    if (prev != NULL) prev->next = newNode;
}
}

```

```

struct Node* readPolynomial() {
    int coeff, exp;
    struct Node* poly = NULL;

    while(1) {
        scanf("%d %d", &coeff, &exp);
        if (coeff== 0 && exp == 0) break;
        insertSorted(&poly,coeff, exp);
    }

    return poly;
}

```

```

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

    while (p1 != NULL) {
        insertSorted(&result, p1->coeff,p1->exp);
        p1 = p1->next;
    }
    while (p2 !=NULL) {
        insertSorted(&result, p2->coeff, p2->exp);
        p2 = p2->next;
    }
    return result;
}

```

```

void displayPolynomial(struct Node* poly) {
    if (poly == NULL) {

```

```
        printf("0\n");
        return;
    }
    while (poly != NULL) {
        printf("%dx^%d", poly->coeff, poly->exp);
        poly = poly-> next;
        if (poly != NULL)
            printf(" + ");
    }
    printf("\n");
}

int main() {
    struct Node* poly1 = readPolynomial();
    struct Node* poly2 = readPolynomial();
    struct Node* sum = addPolynomials(poly1, poly2);

    displayPolynomial(poly1);
    displayPolynomial(poly2);
    displayPolynomial(sum);

    return 0;
}
```

**Status :** Correct

**Marks :** 10/10