# Rajalakshmi Engineering College

Name: Nitin Aakash
Email: 240701370@rajalakshmi.edu.in
Roll no: 240701370
Phone: 9498349045
Branch: REC
Department: I CSE FD
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 : 12.5

### 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 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 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);

    if (degree != 2) {
        printf("Only degree 2 polynomials are supported.\n");
        return 1;
    }

    result = coeff2 * x * x + coeff1 * x + coeff0;
    printf("%d\n", result);

    return 0;
}
```

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

2. 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***

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

struct Node {
    int coeff;
    int exp;
```

```
    struct Node* next;
};

void insertTerm(struct Node** head, int coeff, int exp) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->coeff = coeff;
    newNode->exp = exp;
    newNode->next = NULL;

    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }
}

void printPolynomial(struct Node* head, const char* label) {
    printf("%s", label);
    if (head == NULL) {
        printf("0\n");
        return;
    }

    struct Node* temp = head;
    while (temp != NULL) {
        printf("%dx^%d", temp->coeff, temp->exp);
        if (temp->next != NULL)
            printf(" + ");
        temp = temp->next;
    }
    printf("\n");
}

void simplifyAndPrint(struct Node* head) {
    int simplified[101] = {0};
    struct Node* temp = head;
    while (temp != NULL) {
        simplified[temp->exp] += temp->coeff;
        temp = temp->next;
```

```
    }

    int first = 1;
    for (int i = 100; i >= 0; i--) {
        if (simplified[i] != 0 || (i == 0 && first)) {
            if (!first)
                printf(" + ");
            printf("%dx^%d", simplified[i], i);
            first = 0;
        }
    }
    if (first) {
        printf("0");
    }
    printf("\n");
}

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

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d %d", &coeff, &exp);
        insertTerm(&poly, coeff, exp);
    }

    printPolynomial(poly, "Original polynomial: ");
    printf("Simplified polynomial: ");
    simplifyAndPrint(poly);

    return 0;
}
```

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

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***

```
#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 insertEnd(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 addTerm(struct Node** head, int coeff, int exp) {
    struct Node* temp = *head;
    struct Node* prev = NULL;

    while (temp != NULL && temp->exp > exp) {
        prev = temp;
        temp = temp->next;
    }

    if (temp != NULL && temp->exp == exp) {
        temp->coeff += coeff;
    } else {
        struct Node* newNode = createNode(coeff, exp);
        if (prev == NULL) {
            newNode->next = *head;
            *head = newNode;
        } else {
```

```
            newNode->next = prev->next;
            prev->next = newNode;
        }
    }
}

struct Node* multiply(struct Node* poly1, struct Node* poly2) {
    struct Node* result = NULL;
    struct Node* p1 = poly1;
    struct Node* p2;

    while (p1 != NULL) {
        p2 = poly2;
        while (p2 != NULL) {
            int coeff = p1->coeff * p2->coeff;
            int exp = p1->exp + p2->exp;
            addTerm(&result, coeff, exp);
            p2 = p2->next;
        }
        p1 = p1->next;
    }

    return result;
}

void printPolynomial(struct Node* head) {
    struct Node* temp = head;
    int first = 1;

    while (temp != NULL) {
        if (temp->coeff != 0) {
            if (!first) {
                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);
            }
            first = 0;
```

```
        }
        temp = temp->next;
    }

    if (first) {
        printf("0");
    }

    printf("\n");
}

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

    do {
        scanf("%d %d", &coeff, &exp);
        insertEnd(&poly1, coeff, exp);
        scanf(" %c", &cont);
    } while (cont == 'y' || cont == 'Y');

    do {
        scanf("%d %d", &coeff, &exp);
        insertEnd(&poly2, coeff, exp);
        scanf(" %c", &cont);
    } while (cont == 'y' || cont == 'Y');

    struct Node* result = multiply(poly1, poly2);
    printPolynomial(result);

    return 0;
}
```

***Status :*** Partially correct **Marks : 2.5/10**