NAME:SWETHA.J

ROLL NO:230701357

EX-3: Polynomial Manupilation

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

// Define structure for a term in polynomial
struct Term {
    int coefficient;
    int exponent;
    struct Term *next;
};
typedef struct Term Term;

// Function to create a new term
Term *createTerm(int coeff, int exp) {
    Term *newTerm = (Term *)malloc(sizeof(Term));
    if (newTerm == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newTerm->coefficient = coeff;
    newTerm->exponent = exp;
    newTerm->next = NULL;
    return newTerm;
}

// Function to insert a term into the polynomial
void insertTerm(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;
    }
}

// Function to display the polynomial
void displayPolynomial(Term *poly) {
    if (poly == NULL) {
        printf("Polynomial is empty\n");
    } else {
        while (poly != NULL) {
            printf("(%dx^%d) ", poly->coefficient, poly->exponent);
            poly = poly->next;
            if (poly != NULL) {
                printf("+ ");
```

```
            }
        }
        printf("\n");
    }
}

// Function to add two polynomials
Term *addPolynomials(Term *poly1, Term *poly2) {
    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 {
            insertTerm(&result, poly1->coefficient + poly2-
>coefficient, 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;
}

// Function to subtract two polynomials
Term *subtractPolynomials(Term *poly1, Term *poly2) {
    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 {
            insertTerm(&result, poly1->coefficient - poly2-
>coefficient, 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;
}

// Function to multiply two polynomials
Term *multiplyPolynomials(Term *poly1, Term *poly2) {
    Term *result = NULL;
    Term *temp1 = poly1;
    while (temp1 != NULL) {
        Term *temp2 = poly2;
        while (temp2 != NULL) {
            insertTerm(&result, temp1->coefficient * temp2-
>coefficient, temp1->exponent + temp2->exponent);
            temp2 = temp2->next;
        }
        temp1 = temp1->next;
    }
    return result;
}

// Main function
int main() {
    Term *poly1 = NULL;
    Term *poly2 = NULL;

    // Insert terms for polynomial 1
    insertTerm(&poly1, 5, 2);
    insertTerm(&poly1, -3, 1);
    insertTerm(&poly1, 2, 0);

    // Insert terms for polynomial 2
    insertTerm(&poly2, 4, 3);
    insertTerm(&poly2, 2, 1);

    printf("Polynomial 1: ");
    displayPolynomial(poly1);

    printf("Polynomial 2: ");
    displayPolynomial(poly2);

    Term *sum = addPolynomials(poly1, poly2);
    printf("Sum: ");
    displayPolynomial(sum);

    Term *difference = subtractPolynomials(poly1, poly2);
    printf("Difference: ");
    displayPolynomial(difference);

    Term *product = multiplyPolynomials(poly1, poly2);
    printf("Product: ");
    displayPolynomial(product);

    return 0;
}
```

FOR ADDITION

Enter the values for first polynomial :

Enter the coefficient : 2

Enter the power : 2

Enter 1 to continue : 1

Enter the coefficient : 6

Enter the power : 1

Enter 1 to continue : 1

Enter the coefficient : 5

Enter the power : 0

Enter 1 to continue : 0

The polynomial equation is : 2x^2+6x^1+5x^0

Enter the values for second polynomial :

Enter the coefficient : 3

Enter the power : 2

Enter 1 to continue : 1

Enter the coefficient : -2

Enter the power : 1

Enter 1 to continue : 1

Enter the coefficient : -1

Enter the power : 0

Enter 1 to continue : 0

The polynomial equation is : 3x^2-2x^1-1x^0

The polynomial equation addition result is : 5x^2+4x^1+4x^0

FOR SUBTRACTION

Enter the values for first polynomial :

Enter the coefficient : 3

Enter the power : 2

Enter 1 to continue : 1

Enter the coefficient : 4

Enter the power : 1

Enter 1 to continue : 1

Enter the coefficient : -2

Enter the power : 0

Enter 1 to continue : 0

The polynomial equation is : 3x^2+4x^1-2x^0

Enter the values for second polynomial :

Enter the coefficient : -7

Enter the power : 2

Enter 1 to continue : 1

Enter the coefficient : -10

Enter the power : 1

Enter 1 to continue : 1

Enter the coefficient : 17

Enter the power : 0

Enter 1 to continue : 0

The polynomial equation is : -7x^2-10x^1+17x^0

The polynomial equation subtraction result is : 10x^2+14x^1-19x^0