**WEEK 3**

**Polynomial Manipulation using linked list**

#include <stdio.h>

#include <stdlib.h>

struct Term {

int coefficient;

int exponent;

struct Term \*next;

};

typedef struct Term 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;

}

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;

}

}

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");

}

}

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;

}

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;

}

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;

}

int main() {

Term \*poly1 = NULL;

Term \*poly2 = NULL;

insertTerm(&poly1, 5, 2);

insertTerm(&poly1, -3, 1);

insertTerm(&poly1, 2, 0);

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;

}