Program 09 : Polynomial Evaluation and Addition

Develop a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes

a. Represent and Evaluate a Polynomial $P(x,y,z) = 6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3$

b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)

Support the program with appropriate functions for each of the above operations

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

typedef struct Term {
   int coeff, xExp, yExp, zExp;
   struct Term *next;
} Term;

// Function to create a new term
Term* createTerm(int coeff, int xExp, int yExp, int zExp) {
   Term *newTerm = (Term*)malloc(sizeof(Term));
   newTerm->coeff = coeff;
   newTerm->xExp = xExp;
   newTerm->yExp = yExp;
   newTerm->zExp = zExp;
   newTerm->next = NULL;
   return newTerm;
}

// Function to insert a term in the polynomial
void insertTerm(Term **head, int coeff, int xExp, int yExp, int zExp) {
   Term *newTerm = createTerm(coeff, xExp, yExp, zExp);
   if (*head == NULL) {
      *head = newTerm;
      newTerm->next = *head;
```

```
    } else {
        Term *temp = *head;
        while (temp->next != *head) {
            temp = temp->next;
        }
        temp->next = newTerm;
        newTerm->next = *head;
    }
}

// Function to evaluate a polynomial
double evaluatePolynomial(Term *head, double x, double y, double z) {
    double result = 0.0;
    if (head == NULL) return result;
    Term *temp = head;
    do {
        result += temp->coeff * pow(x, temp->xExp) * pow(y, temp->yExp) *
pow(z, temp->zExp);
        temp = temp->next;
    } while (temp != head);
    return result;
}

// Function to add two polynomials
// Note: This is a simplified version and assumes terms are in the same order
Term* addPolynomials(Term *head1, Term *head2) {
    Term *sumHead = NULL;
    Term *temp1 = head1, *temp2 = head2;
    if (head1 == NULL) return head2;
    if (head2 == NULL) return head1;
    do {
        insertTerm(&sumHead, temp1->coeff + temp2->coeff, temp1->xExp,
temp1->yExp, temp1->zExp);
        temp1 = temp1->next;
        temp2 = temp2->next;
    } while (temp1 != head1 && temp2 != head2);
    return sumHead;
}

int main() {
```

```
  Term *poly1 = NULL, *poly2 = NULL, *polySum = NULL;
  int coeff, xExp, yExp, zExp;
  int n, i;
  double x, y, z, result;

  // Input for POLY1
  printf("Enter the number of terms for POLY1: ");
  scanf("%d", &n);
  for (i = 0; i < n; i++) {
    printf("Enter coeff, x exponent, y exponent, z exponent for term %d: ", i +
1);
    scanf("%d %d %d %d", &coeff, &xExp, &yExp, &zExp);
    insertTerm(&poly1, coeff, xExp, yExp, zExp);
  }

  // Input for POLY2
  printf("Enter the number of terms for POLY2: ");
  scanf("%d", &n);
  for (i = 0; i < n; i++) {
    printf("Enter coeff, x exponent, y exponent, z exponent for term %d: ", i +
1);
    scanf("%d %d %d %d", &coeff, &xExp, &yExp, &zExp);
    insertTerm(&poly2, coeff, xExp, yExp, zExp);
  }

  // Add POLY1 and POLY2
  polySum = addPolynomials(poly1, poly2);

  // Evaluate POLY1
  printf("Enter values for x, y, and z to evaluate POLY1: ");
  scanf("%lf %lf %lf", &x, &y, &z);
  result = evaluatePolynomial(poly1, x, y, z);
  printf("POLY1(%lf, %lf, %lf) = %lf\n", x, y, z, result);

  // Display POLYSUM
  printf("POLYSUM(x, y, z) = ");
  Term *temp = polySum;
  if (temp != NULL) {
    do {
```

```
            printf("%+dx^%dy^%dz ", temp->coeff, temp->xExp, temp->yExp, temp->zExp);
            temp = temp->next;
        } while (temp != polySum);
        printf("\n");
    } else {
        printf("0\n"); // POLYSUM is empty
    }

    // Free memory (not shown, but important in a complete program)

    return 0;
}
```