#include <stdio.h>

#include <stdlib.h>

#include <math.h>

// Node structure for Singly Circular Linked List (SCLL)

struct node {

int coefficient; // Coefficient of the term

int x\_power; // Power of x

int y\_power; // Power of y

int z\_power; // Power of z

struct node\* next; // Pointer to the next node (for circular nature)

};

typedef struct node \*NODEPTR; // Defining NODEPTR as a type for node pointers

// Function prototypes

NODEPTR createNode(int coeff, int x, int y, int z); // Function to create a new node

NODEPTR insertEnd(NODEPTR list, int coeff, int x, int y, int z); // Function to insert a node at the end

void displayPolynomial(NODEPTR list); // Function to display the polynomial

double evaluatePolynomial(NODEPTR list, double x, double y, double z); // Function to evaluate the polynomial

// Create a new node for SCLL with given coefficient and powers of x, y, z

NODEPTR createNode(int coeff, int x, int y, int z)

{

NODEPTR p = (NODEPTR)malloc(sizeof(struct node)); // Allocate memory for a new node

p->coefficient = coeff; // Set the coefficient

p->x\_power = x; // Set the power of x

p->y\_power = y; // Set the power of y

p->z\_power = z; // Set the power of z

p->next = p; // Self-loop for circular nature (next points to itself initially)

return p; // Return the new node

}

// Insert a node at the end of the Singly Circular Linked List (SCLL)

NODEPTR insertEnd(NODEPTR list, int coeff, int x, int y, int z) {

NODEPTR p = createNode(coeff, x, y, z); // Create a new node with given values

if (list == NULL) { // If the list is empty

list = p; // New node becomes the head of the list

} else {

NODEPTR q = list; // Start from the head

while (q->next != list) { // Traverse until we reach the last node (where next points to the head)

q = q->next;

}

q->next = p; // Point the last node to the new node

p->next = list; // Point the new node's next to the head (circular link)

}

return list; // Return the modified list

}

// Display the polynomial represented by the SCLL

void displayPolynomial(NODEPTR list) {

if (list == NULL) { // If the list is empty

printf("Polynomial is empty.\n"); // Print a message and return

return;

}

NODEPTR p = list; // Start from the head of the list

do {

// Print the coefficient and the powers of x, y, z for the current term

printf("%+dx^%dy^%dz^%d ", p->coefficient, p->x\_power, p->y\_power, p->z\_power);

p = p->next; // Move to the next node

} while (p != list); // Stop when we complete the circular traversal (back to the head)

printf("\n"); // Print a new line after the polynomial

}

// Evaluate the polynomial for given values of x, y, and z

double evaluatePolynomial(NODEPTR list, double x, double y, double z)

{

double result = 0.0; // Initialize the result to 0

if (list == NULL) { // If the list is empty

return result; // Return 0 as the result

}

NODEPTR q = list; // Start from the head of the list

do {

// Calculate the value of the term: coefficient \* x^x\_power \* y^y\_power \* z^z\_power

result += q->coefficient \* pow(x, q->x\_power) \* pow(y, q->y\_power) \* pow(z, q->z\_power);

q = q->next; // Move to the next node

} while (q != list); // Stop when we complete the circular traversal

return result; // Return the final result

}

// Main function

int main() {

NODEPTR poly = NULL; // Initialize the polynomial list as NULL (empty)

// Represent the polynomial P(x, y, z) = 6x^2y^2z - 4yz^5 + 3x^3yz + 2xy^5z - 2xyz^3

poly = insertEnd(poly, 6, 2, 2, 1); // Insert term 6x^2y^2z

poly = insertEnd(poly, -4, 0, 1, 5); // Insert term -4yz^5

poly = insertEnd(poly, 3, 3, 1, 1); // Insert term 3x^3yz

poly = insertEnd(poly, 2, 1, 5, 1); // Insert term 2xy^5z

poly = insertEnd(poly, -2, 1, 1, 3); // Insert term -2xyz^3

// Display the polynomial

printf("Polynomial: ");

displayPolynomial(poly); // Display the entire polynomial

// Get the values for x, y, z from the user

double x, y, z;

printf("\nEnter values for x, y, z: ");

scanf("%lf %lf %lf", &x, &y, &z); // Read the values for x, y, z

// Evaluate the polynomial with the entered values

double result = evaluatePolynomial(poly, x, y, z);

printf("Result of evaluation: %.2lf\n", result); // Print the result of the evaluation

return 0; // Return 0 indicating successful execution

}

Sample output

Enter values for x, y, z: 1 2 3

Polynomial: +6x^2y^2z^1 -4x^0y^1z^5 +3x^3y^1z^1 +2x^1y^5z^1 -2x^1y^1z^3

Result of evaluation: ------