Program 9

Design, Develop and Implement 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

Algorithm:

Step 1: Start.
Step 2: Read a polynomial.
Step 3: Represent the polynomial using singly circular linked list. Step 4: Evaluate the given polynomial
Step 5: Read two polynomials and find the sum of the polynomials. Step 6: Stop

Program:

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

/* Node for term: coef * x^px * y^py * z^pz */
struct node {
    int coef;
    int px, py, pz;
    struct node *link;
};

typedef struct node* NODE;

/* Create a header node for circular linked list */
NODE create_header() {
    NODE head = (NODE)malloc(sizeof(struct node));
    if (head == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    head->coef = 0;
    head->px = head->py = head->pz = 0;
    head->link = head;  // circular: header points to itself
    return head;
}

/* Create a new node */
```

```
NODE getnode(int coef, int px, int py, int pz) {
    NODE temp = (NODE)malloc(sizeof(struct node));
    if (temp == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    temp->coef = coef;
    temp->px = px;
    temp->py = py;
    temp->pz = pz;
    temp->link = NULL;
    return temp;
}

/* Insert a term at the end of the circular list */
void insert_end(NODE head, int coef, int px, int py, int pz) {
    if (coef == 0) return; // Ignore zero coefficient terms

    NODE temp = getnode(coef, px, py, pz);

    NODE cur = head;
    while (cur->link != head) {
        cur = cur->link;
    }
    cur->link = temp;
    temp->link = head;
}

/* Insert a term into SUM: if same exponents exist, add the coefficients */
void insert_or_add(NODE head, int coef, int px, int py, int pz) {
    if (coef == 0) return;

    NODE cur = head->link;
    NODE prev = head;

    /* Check if term with same (px,py,pz) already exists */
    while (cur != head) {
        if (cur->px == px && cur->py == py && cur->pz == pz) {
            cur->coef += coef;
            if (cur->coef == 0) {
                /* If coef becomes 0, remove the node */
                prev->link = cur->link;
                free(cur);
            }
            return;
        }
        prev = cur;
        cur = cur->link;
    }

    /* If not found, just insert at end */
    insert_end(head, coef, px, py, pz);
}

/* Display a polynomial */
```

```
void display_poly(NODE head, const char *name) {
    NODE temp = head->link;

    printf("\n%s = ", name);
    if (temp == head) {
        printf("0\n");
        return;
    }

    while (temp != head) {
        printf("%d*x^%d*y^%d*z^%d", temp->coef, temp->px, temp->py, temp->pz);
        temp = temp->link;
        if (temp != head)
            printf(" + ");
    }
    printf("\n");
}

/* Evaluate polynomial for given x, y, z */
long long eval_poly(NODE head, int x, int y, int z) {
    NODE temp = head->link;
    long long sum = 0;

    while (temp != head) {
        long long term = temp->coef;
        term *= (long long)pow(x, temp->px);
        term *= (long long)pow(y, temp->py);
        term *= (long long)pow(z, temp->pz);
        sum += term;
        temp = temp->link;
    }
    return sum;
}

/* Build the given polynomial:
   P(x,y,z) = 6x^2y^2z - 4yz^5 + 3x^3yz + 2xy^5z - 2xyz^3
*/
void build_P(NODE head) {
    insert_end(head,  6, 2, 2, 1); // 6x^2 y^2 z
    insert_end(head, -4, 0, 1, 5); // -4 y z^5
    insert_end(head,  3, 3, 1, 1); // 3 x^3 y z
    insert_end(head,  2, 1, 5, 1); // 2 x y^5 z
    insert_end(head, -2, 1, 1, 3); // -2 x y z^3
}

/* Read a polynomial from user */
void read_poly(NODE head, const char *name) {
    int n, i, coef, px, py, pz;

    /* Clear existing list: make it only header again */
    head->link = head;

    printf("\nEnter number of terms for %s: ", name);
    scanf("%d", &n);
```

```c
    printf("Enter each term as: coef px py pz\n");
    printf("Meaning: coef * x^px * y^py * z^pz\n");

    for (i = 0; i < n; i++) {
        printf("Term %d: ", i + 1);
        scanf("%d%d%d%d", &coef, &px, &py, &pz);
        insert_end(head, coef, px, py, pz);
    }
}

/* Add POLY1 and POLY2 into POLYSUM */
void add_polynomials(NODE poly1, NODE poly2, NODE sum) {
    NODE temp;

    /* Clear old sum */
    sum->link = sum;

    /* Add all terms of poly1 */
    temp = poly1->link;
    while (temp != poly1) {
        insert_or_add(sum, temp->coef, temp->px, temp->py, temp->pz);
        temp = temp->link;
    }

    /* Add all terms of poly2 */
    temp = poly2->link;
    while (temp != poly2) {
        insert_or_add(sum, temp->coef, temp->px, temp->py, temp->pz);
        temp = temp->link;
    }
}

int main() {
    NODE P, POLY1, POLY2, POLYSUM;
    int ch;
    int x, y, z;
    long long value;

    /* Create headers */
    P       = create_header();  // for given P(x,y,z)
    POLY1   = create_header();
    POLY2   = create_header();
    POLYSUM = create_header();

    /* Build fixed P(x,y,z) from question */
    build_P(P);

    while (1) {
        printf("\n------ MENU ------\n");
        printf("1. Display given P(x,y,z)\n");
        printf("2. Evaluate P(x,y,z)\n");
        printf("3. Enter POLY1(x,y,z) and POLY2(x,y,z)\n");
        printf("4. Add POLY1 and POLY2 -> POLYSUM\n");
        printf("5. Display POLY1, POLY2, POLYSUM\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);

        switch (ch) {
        case 1:
            display_poly(P, "P(x,y,z)");
            break;

        case 2:
            printf("Enter x, y, z: ");
            scanf("%d%d%d", &x, &y, &z);
            value = eval_poly(P, x, y, z);
            printf("P(%d,%d,%d) = %lld\n", x, y, z, value);
            break;

        case 3:
            read_poly(POLY1, "POLY1");
            read_poly(POLY2, "POLY2");
            break;

        case 4:
            add_polynomials(POLY1, POLY2, POLYSUM);
            printf("\nPOLYSUM created.\n");
            break;

        case 5:
            display_poly(POLY1, "POLY1(x,y,z)");
            display_poly(POLY2, "POLY2(x,y,z)");
            display_poly(POLYSUM, "POLYSUM(x,y,z)");
            break;

        case 6:
            return 0;

        default:
            printf("Invalid choice. Try again.\n");
        }
    }
}
```