```
// Online C compiler to run C program online
/*Develop a Program in C for the following operation son Singly Circular Linked List (SCLL) with
        header nodes
a. Represent and Evaluate a Polynomial P(x,y,z)=6x2y2z-4yz5+3x3yz+2xy5z-2xyz3
b. Find th esumoftwopolynomialsPOLY1(x,y,z)andPOLY2(x,y,z)andstoretheresultin POLYSUM(x,y,z)
  Support the program with appropriate functions for each of the above operations */



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

// Node structure for a term in the polynomial
 struct PolyTerm{
   int coefficient;
   int pow_x;
   int pow_y;
   int pow_z;
   struct PolyTerm* next;
};

typedef struct PolyTerm* POLYPTR;

POLYPTR fnInsertTerm(POLYPTR poly, int coef, int pow_x, int pow_y, int pow_z)
{
POLYPTR cur;
   POLYPTR newNode = (POLYPTR)malloc(sizeof(struct PolyTerm));
   newNode->coefficient = coef;
   newNode->pow_x = pow_x;
   newNode->pow_y = pow_y;
```

```
    newNode->pow_z = pow_z;
    newNode->next = NULL;

    cur = poly;
    while(cur->next != poly)
    {
    cur = cur->next;
    }
cur->next = newNode;
newNode->next = poly;
return poly;
}

void fnDispPolynomial(POLYPTR poly)
{
    if (poly->next == poly)

{
        printf("Polynomial is empty.\n");
        return;
    }
    POLYPTR cur = poly->next;
    do
    {
        printf("%dx^%dy^%dz^%d ", cur->coefficient, cur->pow_x, cur->pow_y, cur->pow_z);
        cur = cur->next;
        if (cur != poly)
        {
            printf("+ ");
        }
    } while (cur != poly);
    printf("\n");
}

int fnEvaluatePolynomial(POLYPTR poly, int x, int y, int z)
{
    int result = 0;
    if (poly->next == poly)
    {
        return result;
    }
    POLYPTR cur = poly->next;
    do
    {
        int termValue = cur->coefficient;
        termValue *= pow(x, cur->pow_x);
        termValue *= pow(y, cur->pow_y);
        termValue *= pow(z, cur->pow_z);
        result += termValue;
        cur = cur->next;
    } while (cur != poly);
    return result;
}

bool fnMatchTerm(POLYPTR p1, POLYPTR p2)
{
bool bMatches = true;
if(p1->pow_x != p2->pow_x)
bMatches = false;
if(p1->pow_y != p2->pow_y)
bMatches = false;
if(p1->pow_z != p2->pow_z)
```

```
bMatches = false;
return bMatches;
}

POLYPTR fnAddPolynomials(POLYPTR poly1, POLYPTR poly2, POLYPTR polySum)
{
  POLYPTR cur1 = poly1->next;
  POLYPTR cur2 = poly2->next;

  do
  {
  polySum = fnInsertTerm(polySum, cur1->coefficient, cur1->pow_x, cur1->pow_y, cur1->pow_z);
  cur1 = cur1->next;
  }while(cur1 != poly1);
  do
  {
    cur1 = polySum->next;
    bool bMatchFound = false;
    do
    {
      if(fnMatchTerm(cur1, cur2))
      {
        cur1->coefficient += cur2->coefficient;
        bMatchFound = true;
        break;
      }
      cur1 = cur1->next;
    }while(cur1 != polySum);
    if(!bMatchFound)
    {
            polySum = fnInsertTerm(polySum, cur2->coefficient, cur2->pow_x, cur2->pow_y,
cur2->pow_z);
        }
        cur2 = cur2->next;
    }while(cur2 != poly2);
    return polySum;
}
int main()
{
    POLYPTR poly1 = (POLYPTR)malloc(sizeof(struct PolyTerm));
    poly1->next = poly1;
    POLYPTR poly2 = (POLYPTR)malloc(sizeof(struct PolyTerm));
    poly2->next = poly2;
    POLYPTR polySum = (POLYPTR)malloc(sizeof(struct PolyTerm));
    polySum->next = polySum;

 // Represent and evaluate the polynomial P(x, y, z) = 6x^2y^2z • 4yz^5 + 3x^3yz + 2xy^5z • 2xyz^3
    poly1 = fnInsertTerm(poly1, 6, 2, 2, 1);
    poly1 = fnInsertTerm(poly1, 4, 0, 1, 5);
    poly1 = fnInsertTerm(poly1, 3, 3, 1, 1);
    poly1 = fnInsertTerm(poly1, 2, 1, 5, 1);
    poly1 = fnInsertTerm(poly1, 2, 1, 1, 3);
    printf("POLY1(x, y, z) = ");
    fnDispPolynomial(poly1);

    // Read and evaluate the second polynomial POLY2(x, y, z)
    // Represent the polynomial P(x, y, z) = xyz + 4x^3yz
    poly2 = fnInsertTerm(poly2, 1, 1, 1, 1);  // Example term
    poly2 = fnInsertTerm(poly2, 4, 3, 1, 1);

    // Display the second polynomial POLY2(x, y, z)
    printf("POLY2(x, y, z) = ");
    fnDispPolynomial(poly2);

    // Add POLY1(x, y, z) and POLY2(x, y, z) and store the result in POLYSUM(x, y, z)
    polySum = fnAddPolynomials(poly1, poly2, polySum);

    // Display the sum POLYSUM(x, y, z)
    printf("\nPOLYSUM(x, y, z) = ");
    fnDispPolynomial(polySum);

    // Evaluate POLYSUM(x, y, z) for specific values
    int x = 1, y = 2, z = 3;
    int iRes = fnEvaluatePolynomial(polySum, x, y, z);
    printf("\nResult of POLYSUM(%d, %d, %d): %d\n", x, y, z, iRes);

    return 0;
}
```