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

struct node {
    int coef, xexp, yexp, zexp;
    struct node *link;
};
typedef struct node *NODE;

NODE getnode() {
    NODE temp;
    temp = (NODE)malloc(sizeof(struct node));
    if(temp == NULL) {
        printf("Memory not available\n");
        exit(0);
    }
    temp->link = temp;
    return temp;
}

void read_poly(NODE p1, int n) {
    NODE temp, next;
    for(int i = 0; i < n; i++) {
        temp = getnode();
        printf("Enter coefficient and powers (coef xexp yexp zexp): ");
        scanf("%d%d%d%d", &(temp->coef), &(temp->xexp), &(temp->yexp), &(temp->zexp));
        next = p1->link;
        p1->link = temp;
        temp->link = next;
    }
}

void display(NODE p) {
    NODE cur = p->link;
    while(cur != p) {
        printf(" + %d*x^%d*y^%d*z^%d", cur->coef, cur->xexp, cur->yexp, cur->zexp);
        cur = cur->link;
    }
    printf("\n");
}

void insert(NODE p, int coef, int xexp, int yexp, int zexp) {
    NODE next;
    NODE temp = getnode();
    temp->coef = coef;
    temp->xexp = xexp;
    temp->yexp = yexp;
    temp->zexp = zexp;
    next = p->link;
    p->link = temp;
```

```
52      temp->link = next;
53  }
54
55  NODE compare(NODE p, NODE res) {
56      NODE cur;
57      cur = p->link;
58      while(cur != p) {
59          if((cur->xexp == res->xexp) && (cur->yexp == res->yexp) && (cur->zexp
    == res->zexp))
60              return cur;
61          cur = cur->link;
62      }
63      return NULL;
64  }
65
66  void add(NODE p1, NODE p2, NODE p3) {
67      NODE cur, res;
68      cur = p1->link;
69      while(cur != p1) {
70          res = compare(p2, cur);
71          if(res != NULL) {
72              insert(p3, cur->coef + res->coef, cur->xexp, cur->yexp, cur-
    >zexp);
73          } else {
74              insert(p3, cur->coef, cur->xexp, cur->yexp, cur->zexp);
75          }
76          cur = cur->link;
77      }
78      cur = p2->link;
79      while(cur != p2) {
80          if(compare(p1, cur) == NULL) {
81              insert(p3, cur->coef, cur->xexp, cur->yexp, cur->zexp);
82          }
83          cur = cur->link;
84      }
85  }
86
87  void evaluate(NODE p) {
88      int x, y, z, res = 0;
89      NODE cur;
90      printf("Enter value of x, y, and z: ");
91      scanf("%d%d%d", &x, &y, &z);
92      for(cur = p->link; cur != p; cur = cur->link)
93          res = res + cur->coef * pow(x, cur->xexp) * pow(y, cur->yexp) *
    pow(z, cur->zexp);
94      printf("Evaluation of polynomial is %d\n", res);
95  }
96
97  int main() {
98      int n;
99      NODE p1, p2, p3;
100     p1 = getnode();
101     p2 = getnode();
102     p3 = getnode();
103     int ch;
```

```
104     while(1) {
105         printf("1. Evaluate\n2. Polynomial addition\n3. Exit\n");
106         printf("Choice: "); scanf("%d", &ch);
107         switch(ch) {
108             case 1:
109                 printf("Enter the number of terms in P: ");
110                 scanf("%d", &n);
111                 read_poly(p1, n);
112                 printf("Terms in polynomial are ... \n");
113                 display(p1);
114                 evaluate(p1);
115                 break;
116             case 2:
117                 printf("Enter the number of terms in P1: ");
118                 scanf("%d", &n);
119                 read_poly(p1, n);
120                 printf("Enter the number of terms in P2: ");
121                 scanf("%d", &n);
122                 read_poly(p2, n);
123                 printf("Entered Polynomials are:\n");
124                 display(p1);
125                 display(p2);
126                 printf("Polynomial after addition:");
127                 add(p1, p2, p3);
128                 display(p3);
129                 break;
130             default:
131                 exit(0);
132         }
133     }
134 }
135
```