```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

struct node
{
    int coeff;
    int power;
    struct node *next;
};

struct node *create(struct node *start, int z)
{
    struct node *temp, *p;
    printf("Enter the number of terms in polynomial %d: ", z);
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        temp = (struct node *)malloc(sizeof(struct node));
        printf("Enter the coefficient: ");
        scanf("%d", &temp->coeff);
        printf("Enter the power: ");
        scanf("%d", &temp->power);
        temp->next = NULL;
        if (start == NULL)
            start = temp;
        else
        {
            p = start;
            while (p->next != NULL)
                p = p->next;
            p->next = temp;
        }
    }
    return start;
};

void display(struct node *start)
{
    struct node *p;
    p = start;
    while (p != NULL)
    {
        printf("%dx^%d", p->coeff, p->power);
        p = p->next;
        if (p != NULL)
            printf(" + ");
    }
    printf("\n");
```

```c
}

// count number of terms
int count(struct node *start)
{
    struct node *p = start;
    int c = 0;
    while (p != NULL)
    {
        c++;
        p = p->next;
    }
    return c;
}

struct node *add(struct node *start1, struct node *start2)
{
    struct node *start3, *p, *q, *temp;
    start3 = NULL;
    p = start1;
    q = start2;
    while (p != NULL && q != NULL)
    {
        temp = (struct node *)malloc(sizeof(struct node));
        temp->next = NULL;
        if (p->power > q->power)
        {
            temp->coeff = p->coeff;
            temp->power = p->power;
            p = p->next;
        }
        else if (p->power < q->power)
        {
            temp->coeff = q->coeff;
            temp->power = q->power;
            q = q->next;
        }
        else
        {
            temp->coeff = p->coeff + q->coeff;
            temp->power = p->power;
            p = p->next;
            q = q->next;
        }
        if (start3 == NULL)
            start3 = temp;
        else
        {
            struct node *r = start3;
            while (r->next != NULL)
                r = r->next;
            r->next = temp;
        }
    }
    while (p != NULL)
    {
        temp = (struct node *)malloc(sizeof(struct node));
        temp->coeff = p->coeff;
        temp->power = p->power;
        temp->next = NULL;
        p = p->next;
        if (start3 == NULL)
            start3 = temp;
        else
        {
            struct node *r = start3;
            while (r->next != NULL)
                r = r->next;
            r->next = temp;
        }
    }
    while (q != NULL)
    {
        temp = (struct node *)malloc(sizeof(struct node));
        temp->coeff = q->coeff;
        temp->power = q->power;
        temp->next = NULL;
        q = q->next;
        if (start3 == NULL)
            start3 = temp;
        else
        {
            struct node *r = start3;
            while (r->next != NULL)
                r = r->next;
            r->next = temp;
        }
    }
    return start3;
}

struct node *multiply(struct node *start1, struct node *start2)
{
    struct node *start3, *p, *q, *temp;
    start3 = NULL;
    p = start1;
    while (p != NULL)
    {
        q = start2;
        while (q != NULL)
        {
            temp = (struct node *)malloc(sizeof(struct node));
```

```c
            temp->coeff = p->coeff * q->coeff;
            temp->power = p->power + q->power;
            temp->next = NULL;
            if (start3 == NULL)
                start3 = temp;
            else
            {
                struct node *r = start3;
                while (r->next != NULL)
                    r = r->next;
                r->next = temp;
            }
            q = q->next;
        }
        p = p->next;
    }
    return start3;
}

struct node *insertatbeg(struct node *start, int coeff, int power)
{
    struct node *temp;
    temp = (struct node *)malloc(sizeof(struct node));
    temp->coeff = coeff;
    temp->power = power;
    temp->next = start;
    start = temp;
    return start;
}

struct node *insertatend(struct node *start, int coeff, int power)
{
    struct node *temp, *p;
    temp = (struct node *)malloc(sizeof(struct node));
    temp->coeff = coeff;
    temp->power = power;
    temp->next = NULL;
    if (start == NULL)
        start = temp;
    else
    {
        p = start;
        while (p->next != NULL)
            p = p->next;
        p->next = temp;
    }
    return start;
}

// insert at position
struct node *insertatpos(struct node *start, int pos, int coeff, int power)
```

```c
{
    struct node *temp, *p;
    temp = (struct node *)malloc(sizeof(struct node));
    temp->coeff = coeff;
    temp->power = power;
    temp->next = NULL;
    // if beginning, middle or end
    if (pos == 1)
    {
        temp->next = start;
        start = temp;
    }
    else
    {
        p = start;
        for (int i = 1; i < pos - 1; i++)
            p = p->next;
        temp->next = p->next;
        p->next = temp;
    }
    return start;
}

// delete at position
struct node *deleteatpos(struct node *start, int pos)
{
    struct node *p, *q;
    // cheeck if list is empty
    if (start == NULL)
    {
        printf("List is empty");
        return start;
    }
    // if only node
    if (pos == 1)
    {
        p = start;
        start = start->next;
        free(p);
    }
    else
    {
        p = start;
        for (int i = 1; i < pos - 1; i++)
            p = p->next;
        q = p->next;
        p->next = q->next;
        free(q);
    }
    return start;
}
```

```c
struct node *modify(struct node *start)
{
    // display the polynomial first
    printf("\nThe polynomial is: ");
    display(start);
    printf("\nDo You want to \n1.insert \n2.delete a term \n3.Modify a term? \n");
    int ch;
    scanf("%d", &ch);
    if (ch == 1)
    {
        int coeff, power;
        printf("Enter the coefficient: ");
        scanf("%d", &coeff);
        printf("Enter the power: ");
        scanf("%d", &power);
        start = insertatend(start, coeff, power);
    }
    else if (ch == 2)
    {
        printf("Enter the position: ");
        int pos;
        scanf("%d", &pos);
        start = deleteatpos(start, pos);
    }
    else if (ch == 3)
    {
        printf("Enter the position: ");
        int pos;
        scanf("%d", &pos);
        printf("Enter the coefficient: ");
        int coeff;
        scanf("%d", &coeff);
        printf("Enter the power: ");
        int power;
        scanf("%d", &power);
        start = deleteatpos(start, pos);
        start = insertatpos(start, pos, coeff, power);
    }
    else
        printf("Invalid Choice!");
    return start;
}

struct node *inputfromfile(struct node *start, int n)
{
    FILE *fp;
    if (n == 1)
    {
        // create file polynomial1.txt
        fp = fopen("polynomial1.txt", "w");
        printf("Enter the polynomial in the format 4x^3+2x^2+3x^1+5x^0 in file
polynomial1.txt\n");
        fclose(fp);
        // press "k" to continue after adding the polynomial
        printf("Press k to continue\n");
        char ch;
        scanf("%c", &ch);
        scanf("%c", &ch);
        if (ch == 'k')
        {
            // read from file polynomial1.txt
            fp = fopen("polynomial1.txt", "r");
            char ch;
            int coeff, power;
            while (fscanf(fp, "%dx^%d", &coeff, &power) != EOF)
            {
                start = insertatend(start, coeff, power);
                fscanf(fp, "%c", &ch);
            }
            fclose(fp);
        }
    }

    else if (n == 2)
    {
        fp = fopen("polynomial2.txt", "w");
        printf("Enter the polynomial in the format 4x^3+2x^2+3x^1+5x^0 in file
polynomial2.txt\n");
        fclose(fp);
        printf("Press k to continue\n");
        char ch;
        scanf("%c", &ch);
        scanf("%c", &ch);
        if (ch == 'k')
        {
            fp = fopen("polynomial2.txt", "r");
            char ch;
            int coeff, power;
            while (fscanf(fp, "%dx^%d", &coeff, &power) != EOF)
            {
                start = insertatend(start, coeff, power);
                fscanf(fp, "%c", &ch);
            }
            fclose(fp);
        }
    }
    return start;
}

// add like terms of the polynomial
struct node *addliketerms(struct node *start)
```

```c
{
    struct node *p, *q;
    p = start, q = start->next;
    if (q == NULL)
        return start;
    while (q != NULL)
    {
        if (p->power == q->power)
        {
            p->coeff = p->coeff + q->coeff;
            p->next = q->next;
            free(q);
            q = p->next;
        }
        else
        {
            p = q;
            q = q->next;
        }
    }
    return start;
}

// sort the polynomial in descending order of power
struct node *sortll(struct node *start)
{
    struct node *p, *q;
    int temp;
    p = start;
    while (p->next != NULL)
    {
        q = p->next;
        while (q != NULL)
        {
            if (p->power < q->power)
            {
                temp = p->power;
                p->power = q->power;
                q->power = temp;
                temp = p->coeff;
                p->coeff = q->coeff;
                q->coeff = temp;
            }
            q = q->next;
        }
        p = p->next;
    }
    return start;
}

int main()
```

```c
{
    struct node *start1 = NULL, *start2 = NULL;
    while (1)
    {
        printf("\nEnter your choice\n");
        printf("1. Enter Polynomials\n");
        printf("2. Display Polynomials\n");
        printf("3. Add Polynomials\n");
        printf("4. Multiply Polynomials\n");
        printf("5. Modify Polynomials\n");
        printf("6. Take polynomials from file\n");
        printf("7. Exit\n");
        int choice;
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            start1 = create(start1, 1);
            start2 = create(start2, 2);
            break;
        case 2:
            printf("\nPolynomial 1: ");
            display(addliketerms(sortll(start1)));
            printf("\nPolynomial 2: ");
            display(addliketerms(sortll(start2)));
            break;
        case 3:
            printf("\nAddition Result is: ");
            display(addliketerms(sortll(add(start1, start2))));
            break;
        case 4:
            printf("\nMultiplication Result is: ");
            // add like terms of the polynomial after multiplication
            display(addliketerms(sortll(multiply(start1, start2))));
            break;
        case 5:
            printf("Enter the polynomial to modify 1 or 2\n");
            int poly;
            scanf("%d", &poly);
            if (poly == 1)
                start1 = modify(start1);
            else
                start2 = modify(start2);
            break;
        case 6:
            start1 = inputfromfile(start1, 1);
            start2 = inputfromfile(start2, 2);
            break;
        case 7:
            exit(0);
        default:
            printf("Invalid Choice!");
        }
    }
    return 0;
}
```